LogiCORE IP Fast Fourier Transform v7.1

DS260 April 19, 2010 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP Fast Fourier Transform (FFT) implements the Cooley-Tukey FFT algorithm, a computationally efficient method for calculating the Discrete Fourier Transform (DFT).

Features

- Drop-in module for Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA and Spartan-3A/XA/AN/3A DSP FPGAs
- Forward and inverse complex FFT, run-time configurable
- Transform sizes $N = 2^m$, $m = 3 – 16$
- Data sample precision $b_x = 8 – 34$
- Phase factor precision $b_w = 8 – 34$
- Arithmetic types:
  - Unscaled (full-precision) fixed-point
  - Scaled fixed-point
  - Block floating-point
- Fixed-point or floating-point interface
- Rounding or truncation after the butterfly
- Block RAM or Distributed RAM for data and phase-factor storage
- Optional run-time configurable transform point size
- Run-time configurable scaling schedule for scaled fixed-point cores
- Bit/digit reversed or natural output order
- Optional cyclic prefix insertion for digital communications systems
- Four architectures offer a trade-off between core size and transform time
- Bit-accurate C model and MEX function for system modeling available for download
- For use with Xilinx CORE Generator™ software and Xilinx System Generator for DSP v12.1 and higher

Overview

The FFT core computes an N-point forward DFT or inverse DFT (IDFT) where N can be 2^m, $m = 3–16$.

For fixed-point inputs, the input data is a vector of N complex values represented as dual b_x-bit two's-complement numbers, that is, b_x bits for each of the real and imaginary components of the data sample, where b_x is in the range 8 to 34 bits inclusive. Similarly, the phase factors b_w can be 8 to 34 bits wide.

For single-precision floating-point inputs, the input data is a vector of N complex values represented as dual 32-bit floating-point numbers with the phase factors represented as 24- or 25-bit fixed-point numbers.

All memory is on-chip using either block RAM or distributed RAM. The N element output vector is represented using b_y bits for each of the real and imaginary components of the output data. Input data is presented in natural order and the output data can be in either natural or bit/digit reversed order. The complex nature of data input and output is intrinsic to the FFT algorithm, not the implementation.

Three arithmetic options are available for computing the FFT:

- Full-precision unscaled arithmetic
- Scaled fixed-point, where the user provides the scaling schedule
- Block floating-point (run-time adjusted scaling)

The point size N, the choice of forward or inverse transform, the scaling schedule and the cyclic prefix length are run-time configurable. Transform type (forward or inverse), scaling schedule and cyclic prefix length can be changed on a frame-by-frame basis. Changing the point size resets the core.

Four architecture options are available: Pipelined, Streaming I/O, Radix-4, Burst I/O, Radix-2, Burst I/O, and Radix-2 Lite, Burst I/O. For detailed information about each architecture, see "Architecture Options."

Theory of Operation

The FFT is a computationally efficient algorithm for computing a Discrete Fourier Transform (DFT) of sample sizes that are a positive integer power of 2. The DFT $X(k),\ k = 0,\ldots,N-1$ of a sequence $x(n),\ n = 0,\ldots,N-1$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-jnk2\pi/N} \quad k = 0,\ldots,N-1 \qquad \text{Equation 1}$$

where N is the transform size and $j = \sqrt{-1}$. The inverse DFT (IDFT) is given by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{jnk2\pi/N} \quad n = 0,\ldots,N-1 \qquad \text{Equation 2}$$

Algorithm

The FFT core uses the Radix-4 and Radix-2 decompositions for computing the DFT. For Burst I/O architectures, the decimation-in-time (DIT) method is used, while the decimation-in-frequency (DIF) method is used for the Pipelined, Streaming I/O architecture. When using Radix-4 decomposition, the *N*-point FFT consists of $\log_4(N)$ stages, with each stage containing *N/4* Radix-4 butterflies. Point sizes that are not a power of 4 need an extra Radix-2 stage for combining data. An *N*-point FFT using Radix-2 decomposition has $\log_2(N)$ stages, with each stage containing *N/2* Radix-2 butterflies.

The inverse FFT (IFFT) is computed by conjugating the phase factors of the corresponding forward FFT.

Finite Word Length Considerations

The Burst I/O architectures process an array of data by successive passes over the input data array. On each pass, the algorithm performs Radix-4 or Radix-2 butterflies, where each butterfly picks up four or two complex numbers, respectively, and returns four or two complex numbers to the same memory. The numbers returned to memory by the core are potentially larger than the numbers picked up 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; for more information about this topic; see [Ref 1] and [Ref 2].

For a Radix-4 DIT FFT, the values computed in a butterfly stage can experience growth by a factor of up to $1 + 3\sqrt{2} \approx 5.242$. This implies a bit growth of up to 3 bits.

For Radix-2, the growth is by a factor of up to $1+\sqrt{2} \approx 2.414$. This implies a bit growth of up to 2 bits. This bit growth can be handled in three ways:

- Performing the calculations with no scaling and carrying all significant integer bits to the end of the computation
- Scaling at each stage using a fixed-scaling schedule
- Scaling automatically using block floating-point

All significant integer bits are retained when using full-precision unscaled arithmetic. The width of the data path increases to accommodate the bit growth through the butterfly. The growth of the fractional bits created from the multiplication are truncated (or rounded) after the multiplication. The width of the output is (input width + $\log_2$(transform length) + 1). This accommodates the worst case scenario for bit growth.

Consider an unscaled Radix-2 DIT FFT: the data path in each stage must grow by 1 bit as the adder and subtractor in the butterfly may add/subtract two full-scale values and produce a sample which has grown in width by 1 bit. This yields the $\log_2$(transform length) part of the increase in the output width relative to the input width. The complex multiplier preserves the magnitude of an input (as it applies a rotation on the complex plane), but can theoretically produce bit-growth when the magnitude of the input is greater than 1 (for example, 1+j has a magnitude of 1.414). This means that the complex multiplier bit growth must only be considered once in the entire FFT process, yielding the additional +1 increase in the output width relative to the input width. For example, a 1024-point transform with an input of 16 bits consisting of 1 integer bit and 15 fractional bits has an output of 27 bits with 12 integer bits and 15 fractional bits. Note that the core does not have a specific location for the binary point. The output simply maintains the same binary point location as the input. For the preceding example, a 16 bit input with 3 integer bits and 13 fractional bits would have an unscaled output of 27 bits with 14 integer bits and 13 fractional bits.

When using scaling, a scaling schedule is used to divide by a factor of 1, 2, 4, or 8 in each stage. If scaling is insufficient, a butterfly output may grow beyond the dynamic range and cause an overflow. As a result of the scaling applied in the FFT implementation, the transform computed is a scaled transform. The scale factor s is defined as

$$s = 2^{\sum_{i=0}^{\log_2 N-1} b_i} \qquad \text{Equation 3}$$

where b_i is the scaling (specified in bits) applied in stage i.

The scaling results in the final output sequence being modified by the factor $1/s$. For the forward FFT, the output sequence $X'(k)$, $k = 0,...,N-1$ computed by the core is defined as

$$X'(k) = \frac{1}{s}X(k) = \frac{1}{s}\sum_{n=0}^{N-1} x(n)e^{-jnk2\pi/N} \quad k = 0,\ldots,N-1 \qquad \text{Equation 4}$$

For the inverse FFT, the output sequence is

$$x(n) = \frac{1}{s}\sum_{k=0}^{N-1} X(k)e^{jnk2\pi/N} \quad n = 0,\ldots,N-1 \qquad \text{Equation 5}$$

If a Radix-4 algorithm scales by a factor of 4 in each stage, the factor of $1/s$ is equal to the factor of $1/N$ in the inverse FFT equation (Equation 2). For Radix-2, scaling by a factor of 2 in each stage provides the factor of $1/N$.

With block floating-point, each stage applies sufficient scaling to keep numbers in range, and the scaling is tracked by a block exponent.

As with unscaled arithmetic, for scaled and block floating-point arithmetic, the core does not have a specific location for the binary point. The location of the binary point in the output data is inherited from the input data and then shifted by the scaling applied.

Floating-Point Considerations

The FFT core optionally accepts data in IEEE-754 single-precision format with 32-bit words consisting of a 1-bit sign, 8-bit exponent, and 23-bit fraction. The construction of the word matches that of the Xilinx Floating-Point Operator core.

Implementing full floating-point on an FPGA can be expensive in terms of the resources required. The floating-point option in the Xilinx FFT core utilizes a higher precision fixed-point FFT internally to achieve similar noise performance to a full floating-point FFT, with significantly fewer resources. Figure 1 illustrates the two levels of noise performance possible by selecting either 24 bits or 25 bits for the phase factor width. By increasing the phase factor width to 25 bits, more resources may be required, depending on the target FPGA device.

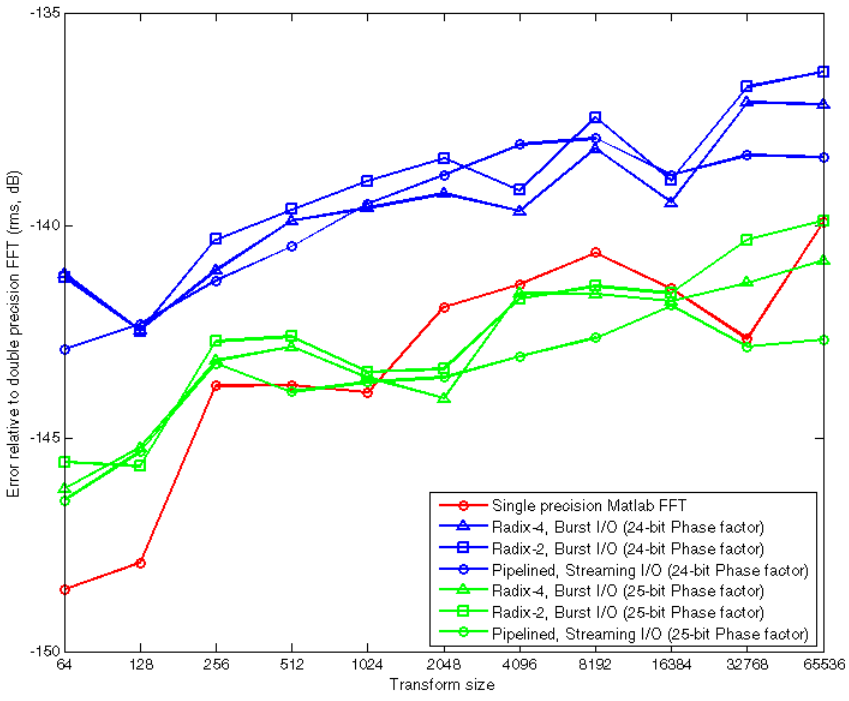


Figure 1: **Comparison of Two Levels of Noise Performance**

Figure 1 shows the ratio of the RMS difference between various models and the double-precision MATLAB® FFT to the data set peak amplitude. The models shown are the single-precision MATLAB FFT function (calculated by casting the input data to single-precision floating-point type), the Xilinx FFT core using a 24-bit phase factor width, and the Xilinx FFT core using a 25-bit phase factor width. To calculate the error signal, a randomized impulse (in magnitude and time) was used as the input signal, with the RMS error averaged over five simulation runs.

All optimization options (memory types and XtremeDSP™ slice optimization) remain available when floating-point input data is selected, allowing the user to trade off resources with transform time.

Transform time for Burst I/O architectures is increased by approximately N, the number of points in the transform, due to the input normalization requirements. For the Pipelined, Streaming I/O architecture, the initial latency to fill the pipeline is increased, but data still streams through the core with no gaps.

Denormalized Numbers

The floating-point interface to the FFT core does not support denormalized numbers. To match the behavior of the Xilinx Floating-Point Operator core, the core treats denormalized operands as zero, with a sign taken from the denormalized number.

NaNs and ± Infinity

If the core detects a NaN or ±Infinity value on the input, all output samples associated with the current input frame are set to NaN. The sign bit is set to zero and all exponent and fraction bits are set to 1.

Real-Valued Input Data

The FFT core accepts complex data samples, but can perform a transform on real-valued data by setting all imaginary input samples to zero.

Due to the finite wordlength effects described previously, noise is introduced during the transform, resulting in the output data not being perfectly symmetric. The DIT and DIF FFT algorithms have different noise effects due to the different calculation order.

For a thorough treatment of this topic, see [Ref 3] and [Ref 4].

The asymmetry between the two halves of the result is more noticeable at larger point sizes. In addition, the noise is more prominent in the lower frequency bins. Therefore, Xilinx recommends that the upper half (N/2+1 to N points) of the output data is used when performing a real-valued FFT.

Rounding Implementation

An option is available, in all architectures, to apply convergent rounding to the data after the butterfly stage. However, selecting this option does not apply convergent rounding to all points in the data path where wordlength reduction occurs.

In particular, the outputs of all complex multipliers in the FFT data path are truncated to reduce data path width (while still maintaining adequate precision) and a simple rounding constant added to the fractional bits. This constant implements non-symmetric, round-towards-minus-infinity rounding, and can introduce a small bias to the FFT results over a large number of samples.

Architecture Options

The FFT core provides four architecture options to offer a trade-off between core size and transform time.

- **"Pipelined, Streaming I/O"** – Allows continuous data processing.
- **"Radix-4, Burst I/O"** – Loads and processes data separately, using an iterative approach. It is smaller in size than the pipelined solution, but has a longer transform time.
- **"Radix-2, Burst I/O"** – Uses the same iterative approach as Radix-4, but the butterfly is smaller. This means it is smaller in size than the Radix-4 solution, but the transform time is longer.
- **"Radix-2 Lite, Burst I/O"** – Based on the Radix-2 architecture, this variant uses a time-multiplexed approach to the butterfly for an even smaller core, at the cost of longer transform time.

Figure 2 illustrates the trade-off of throughput versus resource use for the four architectures. As a rule of thumb, each architecture offers a factor of 2 difference in resource from the next architecture. The example is for an even power of 2 point size. This does not require the Radix-4 architecture to have an additional Radix-2 stage.

All four architectures may be configured to use a fixed-point interface with one of three fixed-point arithmetic methods (unscaled, scaled or block floating-point) or may instead use a floating-point interface.

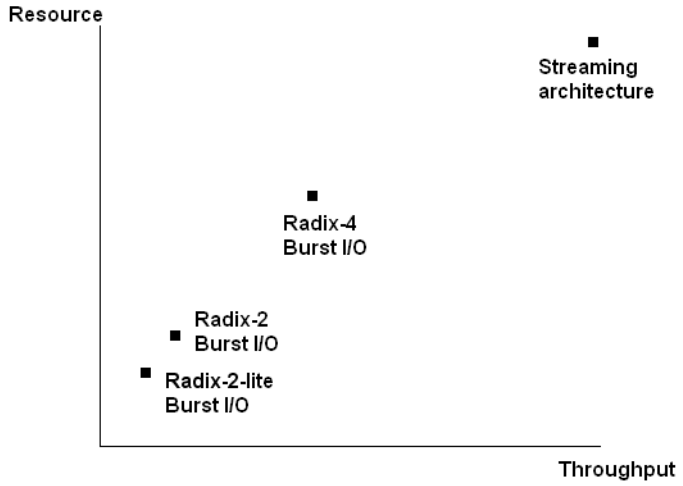


Figure 2: **Resource versus Throughput for Architecture Options**

Bit and Digit Reversal

Each architecture offers the option of natural or reversed ordering of output data, with data being input in natural order. The FFT algorithm reorders the samples during processing such that data input in natural order is output in reversed order. The core can optionally output the data in natural order. However, this imposes a cost on each architecture. For the Burst I/O architectures, this imposes a time penalty, because unloading the data cannot take place at the same time as loading input data for the next frame, so separate unload and load phases are required. In the pipelined architecture, it requires additional RAM storage to perform the reordering.

In the Radix-2, Burst I/O, Radix-2 Lite, Burst I/O, and Pipelined, Streaming I/O architectures, the Bit Reverse order is simple to calculate, by taking the index of the data point, written in binary, and reversing the order of the digits. Hence, 0000, 0001, 0010, 0011, 0100,...(0, 1, 2, 3, 4,...) becomes 0000, 1000, 0100, 1100, 0010,...(0, 8, 4, 12, 2,...).

In the case of the Radix-4, Burst I/O architecture, the reversal applies to *digits* and, therefore, is called Digit Reversal. A digit in Radix-4 is two bits. Hence, 0000, 0001, 0010, 0011, 0100,...(0, 1, 2, 3, 4,...) becomes 0000, 0100, 1000, 1100, 0001,...(0, 4, 8, 12, 1,...), as the pairs of digits are reversed. Where the transform size requires an odd number of index bits, the odd digit in the least significant place is moved to the most significant place, so 00000, 00001, 00010, 00011, 00100,... (0, 1, 2, 3, 4,...) becomes 00000, 10000, 00100, 10100, 01000,...(0, 16, 4, 20, 8,...)

Note: The core outputs a data point index along with the data, so this section is for information only.

Pipelined, Streaming I/O

The Pipelined, Streaming I/O solution pipelines several Radix-2 butterfly processing engines to offer continuous data processing. Each processing engine has its own memory banks to store the input and intermediate data (Figure 3). The core has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next frame of data, and unload the results of the previous frame of data. The user can continuously stream in data and, after the calculation latency, can continuously unload the results. If preferred, this design can also calculate one frame by itself or frames with gaps in between.

In the scaled fixed-point mode, the data is scaled after every pair of Radix-2 stages. The block floating-point mode may use significantly more resources than the scaled mode, as it must maintain extra bits of precision to allow dynamic scaling without impacting performance. Therefore, if the input data is well understood and is unlikely to exhibit large amplitude fluctuation, using scaled arithmetic (with a suitable scaling schedule to avoid overflow in the known worst case) is sufficient, and resources may be saved.

The input data is presented in natural order. The unloaded output data can either be in bit reversed order or in natural order. When natural order output data is selected, additional memory resource is utilized.

This architecture covers point sizes from 8 to 65536. The user has flexibility to select the number of stages to use block RAM for data and phase factor storage. The remaining stages use distributed memory.

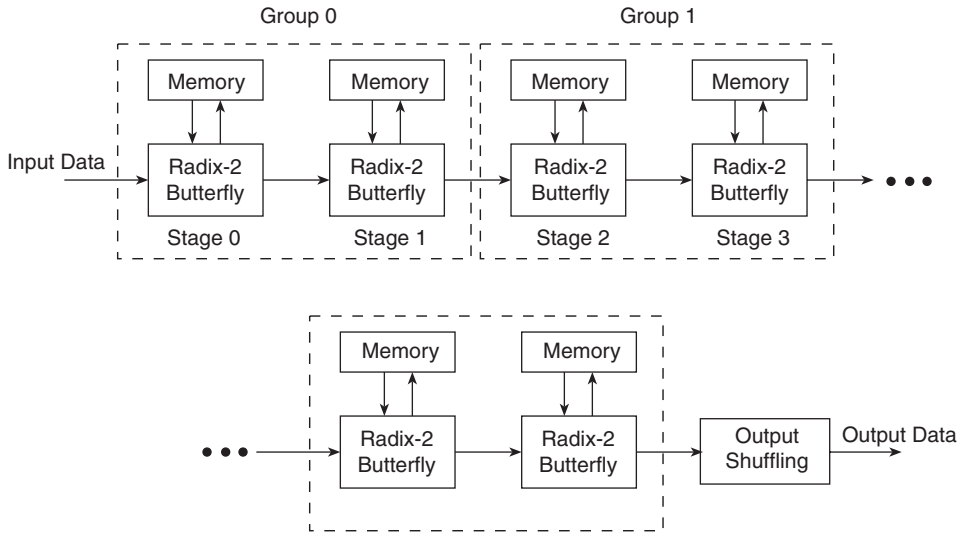


Figure 3: **Pipelined, Streaming I/O**

Radix-4, Burst I/O

With the Radix-4, Burst I/O solution, the FFT core uses one Radix-4 butterfly processing engine (Figure 4). It loads and/or unloads data separately from calculating the transform. Data I/O and processing are not simultaneous. When the FFT is started, the data is loaded. After a full frame has been loaded, the core computes the transform. When the computation has finished, the data can be unloaded, but cannot be loaded or unloaded during the calculation process. The data loading and unloading processes can be overlapped if the data is unloaded in digit reversed order.

This architecture has lower resource usage than the Pipelined, Streaming I/O architecture, but a longer transform time, and supports point sizes from 64 to 65536. Data and phase factors can be stored in block RAM or in distributed RAM (the latter for point sizes less than or equal to 1024).

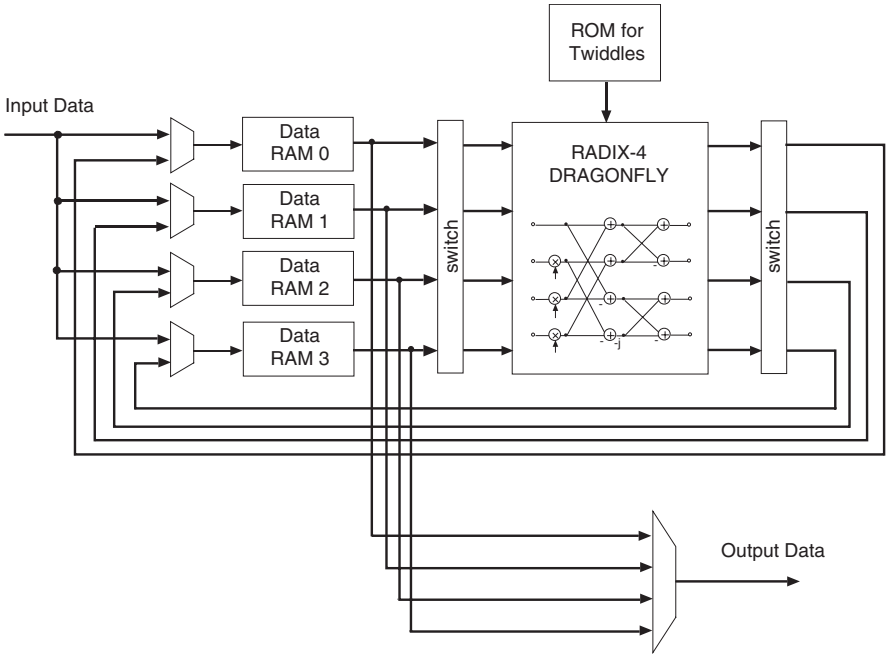


Figure 4: **Radix-4, Burst I/O**

Radix-2, Burst I/O

The Radix-2, Burst I/O architecture uses one Radix-2 butterfly processing engine (Figure 5). After a frame of data is loaded, the input data stream must halt until the transform calculation is completed. Then, the data can be unloaded. As with the Radix-4, Burst I/O architecture, data can be simultaneously loaded and unloaded when the output samples are in bit reversed order. This solution supports point sizes from 8 to 65536. Both the data memories and phase factor memories can be in either block RAM or distributed RAM (the latter for point sizes less than or equal to 1024).

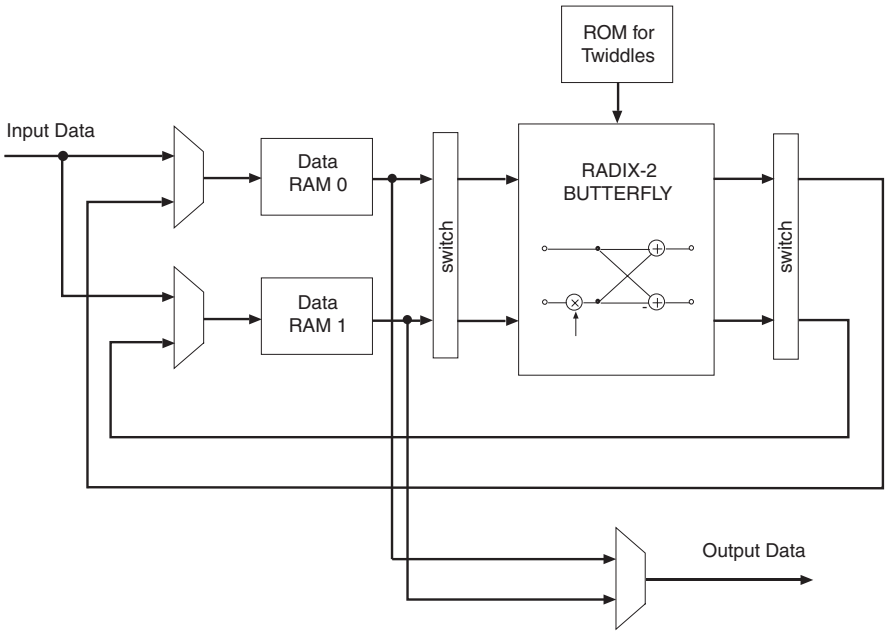


Figure 5: **Radix-2, Burst I/O**

Radix-2 Lite, Burst I/O

This architecture differs from the Radix-2, Burst I/O in that the butterfly processing engine uses one shared adder/subtractor, hence reducing resources at the expense of an additional delay per butterfly calculation. Again, as with the Radix-4 and Radix-2, Burst I/O architectures, data can be simultaneously loaded and unloaded only if the output samples are in bit reversed order. This solution supports point sizes from 8 to 65536. See Figure 6.

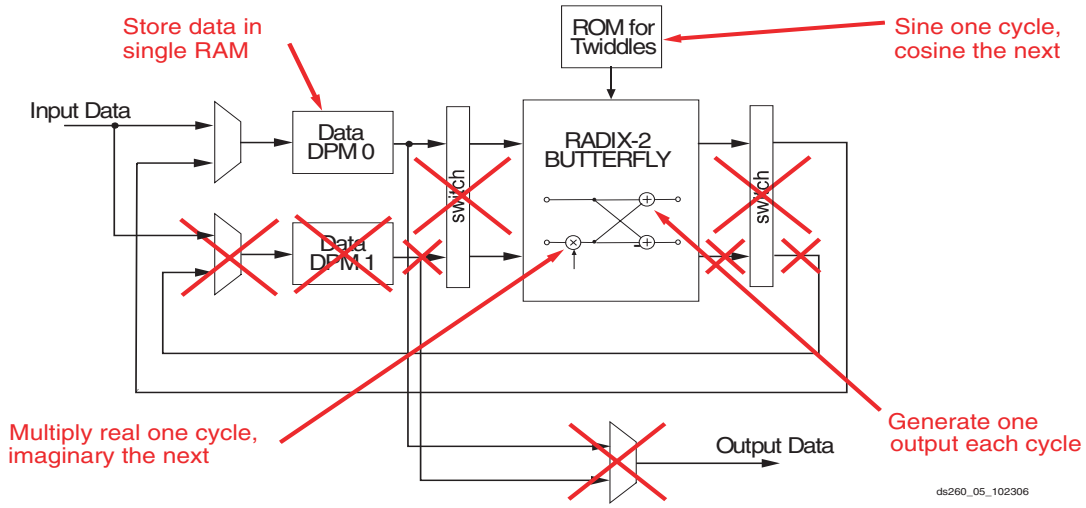


Figure 6: **Radix-2 Lite, Burst I/O**

Core Symbol and Port Definitions

Figure 7 shows the Core Schematic Symbol, and Table 1 lists the core pinout for single channel configurations.

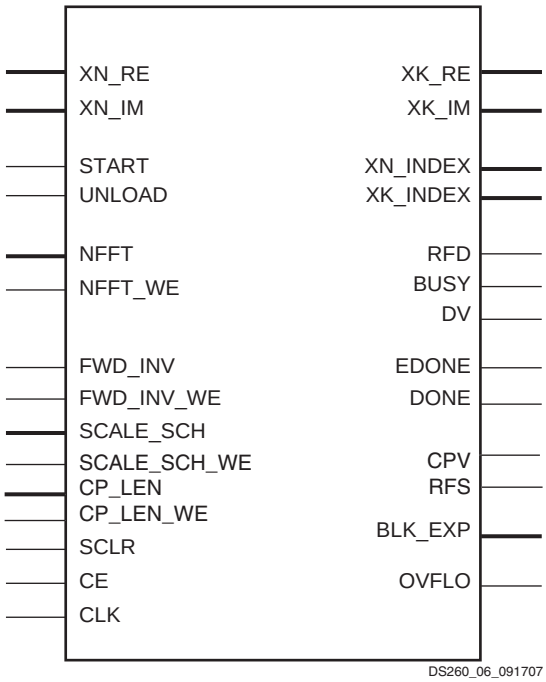


Figure 7: **Core Schematic Symbol (Single Channel)**

Table 1: **Core Pinout (Single Channel)**

Port Name	Port Width	Direction	Description
XN_RE	b_{xn}	Input	Input data bus: Real component (b_{xn} = 8 - 34) in two's complement or single precision floating-point format.
XN_IM	b_{xn}	Input	Input data bus. Imaginary component (b_{xn} = 8 - 34) in two's complement or single precision floating-point format.
START	1	Input	FFT start signal (Active High): START is asserted to begin the data loading and transform calculation (for the Burst I/O architectures). For Streaming I/O, START begins data loading, which proceeds directly to transform calculation and then data unloading.
UNLOAD	1	Input	Result unloading (Active High): For the Burst I/O architectures, UNLOAD starts the unloading of the results in natural order. The UNLOAD port is not necessary for the Pipelined, Streaming I/O architecture or for bit/digit reversed unloading.
NFFT	5	Input	Point size of the transform: NFFT can be the size of the transform or any smaller point size. For example, a 1024-point FFT can compute point sizes 1024, 512, 256, and so on. The value of NFFT is $\log_2$ (point size). This port is only used with run-time configurable transform point size.
NFFT_WE	1	Input	Write enable for NFFT (Active High): Asserting NFFT_WE causes the core to stop all processes and to initialize the state of the core to the new point size on the NFFT port. This port is only used with run-time configurable transform point size. Clock Enable overrides NFFT_WE if both signals are present.
FWD_INV	1	Input	Control signal that indicates if a forward FFT or an inverse FFT is performed. When FWD_INV = 1, a forward transform is computed. If FWD_INV = 0, an inverse transform is computed.
FWD_INV_WE	1	Input	Write enable for FWD_INV (Active High).
SCALE_SCH	$2 \times ceil\left(\frac{NFFT}{2}\right)$ for PIpelined, Streaming I/O and Radix-4, Burst I/O architectures or 2 x $NFFT$ for Radix-2, Burst I/O and Radix-2 Lite, Burst I/O architectures where $NFFT$ is $\log_2$ (maximum point size) or the number of stages	Input	Scaling schedule: For Burst I/O architectures, the scaling schedule is specified with two bits for each stage, with the scaling for the first stage given by the two LSBs. The scaling can be specified as 3, 2, 1, or 0, which represents the number of bits to be shifted. An example scaling schedule for N =1024, Radix-4, Burst I/O is [1 0 2 3 2] (ordered from last to first stage). For N =128, Radix-2, Burst I/O or Radix-2 Lite, Burst I/O, one possible scaling schedule is [1 1 1 1 0 1 2] (ordered from last to first stage). For Pipelined, Streaming I/O architecture, the scaling schedule is specified with two bits for every pair of Radix-2 stages, starting at the two LSBs. For example, a scaling schedule for N = 256 could be [2 2 2 3]. When N is not a power of 4, the maximum bit growth for the last stage is one bit. For instance, [0 2 2 2 2] or [1 2 2 2 2] are valid scaling schedules for N = 512, but [2 2 2 2 2] is invalid. For this transform length. the two MSBs of SCALE_SCH can only be 00 or 01. This port is only available with scaled arithmetic (not unscaled, block floating-point or single precision floating-point).

Table 1: **Core Pinout (Single Channel) *(Cont'd)***

Port Name	Port Width	Direction	Description
SCALE_SCH_WE	1	Input	Write enable for SCALE_SCH (Active High): This port is available only with scaled arithmetic.
CP_LEN	$\log_2$ (maximum point size)	Input	Cyclic prefix length: The number of samples from the end of the transform that are initially output as a cyclic prefix, before the whole transform is output. CP_LEN can be any number from zero to one less than the point size. This port is only available with cyclic prefix insertion.
CP_LEN_WE	1	Input	Write enable for CP_LEN (Active High): This port is only available with cyclic prefix insertion.
SCLR	1	Input	Master synchronous reset (Active High): Optional port. The synchronous reset overrides clock enable when both are present on the core.
CE	1	Input	Clock enable (Active High): Optional port.
CLK	1	Input	Rising-edge clock
XK_RE	b_{xk}	Output	Output data bus: Real component in two's complement or floating-point format. (For scaled arithmetic and block floating-point arithmetic, $b_{xk} = b_{xn}$. For unscaled arithmetic, $b_{xk} = b_{xn} + \log_2$ (maximum point size) $+1$. For single precision floating-point $b_{xk} = 32$).
XK_IM	b_{xk}	Output	Output data bus: Imaginary component in two's complement or single precision floating-point format. (For scaled arithmetic and block floating-point arithmetic, $b_{xk} = b_{xn}$. For unscaled arithmetic, $b_{xk} = b_{xn} + \log_2$ (maximum point size) $+1$. For single precision floating-point $b_{xk} = 32$)
XN_INDEX	$\log_2$ (maximum point size)	Output	Index of input data.
XK_INDEX	$\log_2$ (maximum point size)	Output	Index of output data.
RFD	1	Output	Ready for data (Active High): RFD is High during the load operation.
BUSY	1	Output	Core activity indicator (Active High): This signal goes High while the core is computing the transform.
DV	1	Output	Data valid (Active High): This signal is High when valid data is presented at the output.
EDONE	1	Output	Early done strobe (Active High): EDONE goes High one clock cycle immediately prior to DONE going High.
DONE	1	Output	FFT complete strobe (Active High): DONE transitions High for one clock cycle when the transform calculation has completed.
BLK_EXP	5	Output	Block exponent: The amount of scaling applied. Available only when block floating-point is used.
OVFLO	1	Output	Arithmetic overflow indicator (Active High): OVFLO is High during result unloading if any value in the data frame overflowed. The OVFLO signal is reset at the beginning of a new frame of data. This port is optional and only available with scaled arithmetic or single precision floating-point I/O.

Table 1: **Core Pinout (Single Channel) *(Cont'd)***

Port Name	Port Width	Direction	Description
CPV	1	Output	Cyclic prefix valid (Active High): This signal is High when valid data that is part of the cyclic prefix is presented at the output. This port is only available with cyclic prefix insertion.
RFS	1	Output	Ready for start (Active High): This signal goes High when the core is ready to accept an assertion on the START input to begin data loading. This port is only available with cyclic prefix insertion and the Pipelined, Streaming I/O architecture.

Multichannel Pinout

Up to 12 channels are supported, for Burst I/O architectures only. Table 2 shows how the preceding pinout must be adapted for multichannel operation.

Table 2: **Single to Multichannel Pinout Conversion**

Single Channel	Multichannel
CLK	CLK
CE	CE
SCLR	SCLR
NFFT	NFFT
NFFT_WE	NFFT_WE
FWD_INV	FWD_INV0,.,FWD_INV11
FWD_INV_WE	FWD_INV0_WE,.,FWD_INV11_WE
START	START
UNLOAD	UNLOAD
XN_RE	XN0_RE,.,XN11_RE
XN_IM	XN0_IM,.,XN11_IM
SCALE_SCH	SCALE_SCH0,.,SCALE_SCH11
SCALE_SCH_WE	SCALE_SCH0_WE,.,SCALE_SCH11_WE
CP_LEN	CP_LEN
CP_LEN_WE	CP_LEN_WE
RFD	RFD
XN_INDEX	XN_INDEX
BUSY	BUSY
EDONE	EDONE
DONE	DONE
DV	DV
XK_INDEX	XK_INDEX
XK_RE	XK0_RE,.,XK11_RE
XK_IM	XK0_IM,.,XK11_IM
BLK_EXP	BLK_EXP0,.,BLK_EXP11
OVFLO	OVFLO0,.,OVFLO11

Table 2: **Single to Multichannel Pinout Conversion *(Cont'd)***

Single Channel	Multichannel
CPV	CPV
RFS	RFS

CORE Generator Graphical User Interface

The FFT core graphical user interface (GUI) provides several screens with fields to set the parameter values for the particular instantiation required. A description of each CORE Generator GUI field follows:

Page 1

- **Component Name**: The name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9, and "_".
- **Channels**: Select the number of channels from 1 to 12. Multichannel operation is available for the three Burst I/O architectures.
- **Transform Length**: Select the desired point size. All powers of two from 8 to 65536 are available.
- **Implementation Options**: Select an implementation option, as described in "Architecture Options," page 6.
  - The Pipelined, Streaming I/O, Radix-2, Burst I/O, and Radix-2 Lite, Burst I/O architectures support point sizes 8 to 65536.
  - The Radix-4, Burst I/O architecture supports point sizes 64 to 65536.
  - Check Automatically Select to choose the smallest implementation that meets the specified Target Data Throughput, provided the specified Target Clock Frequency is achieved when the FFT core is implemented on an FPGA device.
  - Target Clock Frequency and Target Data Throughput are only used to automatically select an implementation and to calculate latency. The core is not guaranteed to run at the specified target clock frequency or target data throughput.
- **Transform Length Options**: Select the transform length to be run-time configurable or not. The core uses fewer logic resources and has a faster maximum clock speed when the transform length is not run-time configurable.

Page 2

- **Data Format**: Select whether the input and output data samples are in Fixed Point format, or in IEEE-754 single precision (32-bit) Floating-Point format. Floating-Point format is not available when the core is in a multichannel configuration.
- **Precision Options**: Input data and phase factors can be independently configured to widths from 8 to 34 bits, inclusive. When the Data Format is Floating-Point, the input data width is fixed at 32 bits and the phase factor width can be set to 24 or 25 bits depending on the noise performance required and available resources.
- **Scaling Options**: Three options are available, for all architectures:
  - Unscaled
    - All integer bit growth is carried to the output. This can use more FPGA resources.
  - Scaled
    - A user-defined scaling schedule determines how data is scaled between FFT stages.

- Block Floating-Point
  - The core determines how much scaling is necessary to make best use of available dynamic range, and reports the scaling factor as a block exponent.

- **Optional Pins**: Clock Enable (CE), Synchronous Clear (SCLR), and Overflow (OVFLO) are optional pins. Synchronous Clear overrides Clock Enable if both are selected. If an option is not selected, some logic resources may be saved and a higher clock frequency may be attainable.
- **Rounding Modes**: At the output of the butterfly, the LSBs in the data path need to be trimmed. These bits can be truncated or rounded using convergent rounding, which is an unbiased rounding scheme. When the fractional part of a number is equal to exactly one-half, convergent rounding rounds up if the number is odd, and rounds down if the number is even. Convergent rounding can be used to avoid the DC bias that would otherwise be introduced by truncation after the butterfly stages. Selecting this option will increase slice usage and yields a small increase in transform time due to additional latency.
- **Output Ordering**: Output data selections are either Bit/Digit Reversed Order or Natural Order. The Radix-2 based architectures (Pipelined, Streaming I/O, Radix-2, Burst I/O and Radix-2 Lite, Burst I/O) offer bit-reversed ordering, and the Radix-4 based architecture (Radix-4, Burst I/O) offers digit-reversed ordering. For the Pipelined, Streaming I/O architecture, selecting natural order output ordering results in an increase in memory used by the core. For Burst I/O architectures, selecting natural order output increases the overall transform time because a separate unloading phase is required.
  - Cyclic Prefix Insertion can be selected if the output ordering is Natural Order. Cyclic Prefix Insertion is available for all architectures, and is typically used in OFDM wireless communications systems.
- **Input Data Timing**: In previous versions of the Xilinx FFT core, input data was applied three cycles after the corresponding sample index to allow a block memory containing data samples to be addressed. In many cases, this was not necessary, and applying data on the wrong cycle made it appear the core was functioning incorrectly. This timing may now be configured to be backwards-compatible with previous versions, or have no delay between sample index and applied data (default).

Page 3

- **Memory Options**:
  - **Data And Phase Factors (Burst I/O architectures)**: For Burst I/O architectures, either block RAM or distributed RAM can be used for data and phase factor storage. Data and phase factor storage can be in distributed RAM for all point sizes up to and including 1024 points.
  - **Data And Phase Factors (Pipelined, Streaming I/O)**: In the Pipelined, Streaming I/O solution, the data can be stored partially in block RAM and partially in distributed RAM. Each pipeline stage, counting from the input side, uses smaller data and phase factor memories than preceding stages. The user can select the number of pipeline stages that use block RAM for data and phase factor storage. Later stages use distributed RAM. The default displayed on the GUI offers a good balance between both. If output ordering is Natural Order, the memory used for the reorder buffer can be either block RAM or distributed RAM. The reorder buffer can use distributed RAM for point sizes less than or equal to 1024.
    - When block floating-point is selected for the Pipelined, Streaming I/O architecture, a RAM buffer is required for natural order *and* bit reversed order output data. In this case, the reorder buffer options remain available and distributed RAM may be selected for all point sizes below 2048.

- **Hybrid Memories**: Where data, phase factor, or reorder buffer memories are stored in block RAM, if the size of the memory is greater than one block RAM, the memory can be constructed from a hybrid of block RAMs and distributed RAM, where the majority of the data is stored in block RAMs and a few bits that are left over are stored in distributed RAM. This Hybrid Memory is an alternative to constructing the memory entirely from multiple block RAMs. It provides a reduction in the block RAM count, at the cost of an increase in the number of slices used. Hybrid Memories are only available when block RAM is used for one or more memories and the number of slices required for a Hybrid Memory implementation is below an internal threshold of 256 LUTs per memory. If these conditions are met, Hybrid Memories are made available and can be selected.

- **Optimize Options**:
  - **Complex Multipliers**: Three options are available for customization of the complex multiplier implementation:
    - **Use CLB logic**: All complex multipliers will be constructed using slice logic. This is appropriate for target applications that have low performance requirements, or target devices that have few XtremeDSP slices/Mult18x18s.
    - **Use 3-multiplier structure (resource optimization)**: All complex multipliers will use a three real multiply, five add/subtract structure, where the multipliers use XtremeDSP slices/Mult18x18s. This reduces the XtremeDSP slice/Mult18x18 count, but uses some slice logic. In Spartan-3A DSP, Spartan-6, and Virtex-6 devices, this structure can make use of the XtremeDSP slice pre-adder to reduce or remove the need for extra slice logic, and improve performance.
    - **Use 4-multiplier structure (performance optimization)**: All complex multipliers will use a four real multiply, two add/subtract structure, utilizing XtremeDSP slices/Mult18x18s. This structure yields the highest clock performance at the expense of more dedicated multipliers. In devices with XtremeDSP slices, the add/subtract operations are implemented within the XtremeDSP slices. In devices with Mult18x18s, the add/subtract operations use slice logic.

    Note: The core may override the complex multiplier implementation internally to ensure the fewest number of XtremeDSP slices/Mult18x18s are used, without impacting performance. For this reason, some core configurations may show no difference in XtremeDSP slice/Mult18x18 usage when toggling between the 3-multiplier and 4-multiplier options. If “Use CLB logic” is selected, however, slice logic will always be utilized.
  - **Butterfly Arithmetic**: Two options are available for customization of the butterfly implementation:
    - **Use CLB logic**: All butterfly stages will be constructed using slice logic.
    - **Use XtremeDSP Slices:** For devices with XtremeDSP slices, this option forces all butterfly stages to be implemented using the adder/subtracters in XtremeDSP slices.

Information Tabs

- **Resource Estimates**:
  - **Implementation**: This field displays the currently selected architecture. This is useful to see the result of automatic architecture selection.
  - **Transform Size**: When the transform length is run-time configurable, the core has the ability to reprogram the point size while the core is running; that is, the core can support the selected point size and any smaller point size. This field displays the supported point sizes based on the Transform Length, Transform Length Options, and the Implementation Options selected.
  - **Output Data Width**: The output data width equals the input data width for scaled arithmetic and block floating-point arithmetic. With unscaled arithmetic, the output data width equals (input data width + $\log_2$(point size) + 1).
  - **Resource Estimates**: Based on the options selected, this field displays the XtremeDSP slice or Mult18x18 count and 18K block RAM numbers (9K block RAM numbers for Spartan-6 devices). The resource numbers are just an estimate. For exact resource usage, and slice/LUT-FlipFlop pair information, a MAP report should be consulted.
- **Latency:**
  - This tab shows the latency of the FFT core in clock cycles and microseconds (µs) for each point size supported. The latency is from asserting the `START` input to the last sample of output data coming out of the core, assuming that the `UNLOAD` input (if present) is asserted as soon as `DONE` goes High. Note that this is not the minimum number of cycles between starting consecutive frames, as frames may overlap in some cases. The latency in microseconds is based on the target clock frequency. The latency figures can be copied to the Clipboard and pasted as plain text into other applications.
- **C Model**:
  - This tab provides a link to the Xilinx LogiCORE IP FFT web page where the core C model can be downloaded. For details of the C model, see "Bit-Accurate C Model," page 22.

XCO Parameters

Table 3 defines valid entries for the XCO parameters. Parameters are not case sensitive. Default values are displayed in bold. Xilinx strongly recommends that XCO parameters are not manually edited in the XCO file; instead, use the CORE Generator GUI to configure the core and perform range and parameter value checking.

Table 3: **XCO Parameters**

XCO Parameter	Valid Values
component_name	Name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9, and "_".
channels	1 - 12 (default value is **1**)
transform_length	8, 16, 32, 64, 128, 256, 512, **1024**, 2048, 4096, 8192, 16384, 32768, 65536
implementation_options	automatically_select pipelined_streaming_io radix_4_burst_io radix_2_burst_io radix_2_lite_burst_io
target_clock_frequency	0 - 550 (default is **250**)

Table 3: **XCO Parameters *(Cont'd)***

XCO Parameter	Valid Values
target_data_throughput	0 - 550 (default is **50**)
run_time_configurable_transform_length	false true
data_format	fixed_point floating_point
input_width	8 - 34 (default value is **16**)
phase_factor_width	8 - 34 (default value is **16**)
scaling_options	scaled unscaled block_floating_point
rounding_modes	truncation convergent_rounding
ce	false true
sclr	false true
ovflo	false true
output_ordering	bit_reversed_order natural_order
cyclic_prefix_insertion	false true
memory_options_data	block_ram distributed_ram
memory_options_phase_factors	block_ram distributed_ram
memory_options_reorder	block_ram distributed_ram
number_of_stages_using_block_ram_for_data_and_phase_factors	0 - 11 (default value depends on transform length)
memory_options_hybrid	false true
input_data_offset	no_offset three_cycle_offset
complex_mult_type	use_luts use_mults_resources use_mults_performance
butterfly_type	use_luts use_xtremedsp_slices

Simulation Models

When the core is generated using the CORE Generator software, a UniSim-based simulation model is created. The FFT core does not have a VHDL or Verilog functional behavioral model. For this reason, the core overrides the CORE Generator Project Options and always delivers a Structural model type.

Xilinx recommends that the designer run simulations using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution to work properly in either functional or timing simulation. The FFT core UniSIm-based structural model may produce incorrect results if simulated with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in Chapter 6 of the *Synthesis and Simulation Design Guide* for more information. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

Migrating to FFT v7.1 from Earlier Versions

The CORE Generator core update feature many be used to update an existing FFT XCO file to version 7.1 of the FFT core. The core may then be regenerated to create a new netlist. See the CORE Generator documentation for more information on this feature.

Port Changes

Ports added in later versions of the FFT are incremental additions, and will not appear in a new netlist created by using the core update feature described previously.

There are no differences in port naming conventions, polarities, priorities or widths between versions.

Latency and Resource Changes

The latency and resource usage of FFT v7.1 may differ from previous versions. To establish how these properties have changed, if at all, the updated XCO file may be loaded into CORE Generator and the Latency and Resource Estimate tabs on the left side of the GUI inspected. The core should be run through map to establish if LUT-FF pair (or slice) utilization has changed.

Updating from Versions before FFT v5.0

In FFT v5.0, additional rounding was added after the complex multipliers in the data path to improve the numerical results, and hence signal-to-noise ratio. This rounding is always present, and is not influenced by the "convergent rounding" parameter.

The impact of this change is that FFT core versions v5.0 and later are no longer bit-accurate to FFT v4.1 and earlier versions. In many cases, this difference appears only in the LSBs of the output.

System Generator For DSP Graphical User Interface

This section describes each tab of the System Generator GUI and details the parameters that differ from the CORE Generator GUI. See "CORE Generator Graphical User Interface" for more detailed information about all other parameters.

Tab 1: Basic

The Basic tab is used to specify the transform configuration and architecture in a similar way to page 1 of the CORE Generator GUI.

Implementation Options: Select an implementation option as described in "Architecture Options."

- The Pipelined, Streaming I/O, Radix-2, Burst I/O, and Radix-2 Lite, Burst I/O architectures support point sizes 8 to 65536.
- The Radix-4, Burst I/O architecture supports point sizes 64 to 65536.

System Generator supports only single-channel implementation of the FFT and, hence, Channels is not available as a GUI option.

Tab 2: Advanced

The Advanced tab is used to specify phase factor precision, scaling, rounding, and optional port options in a similar way to page 2 of the CORE Generator GUI.

- **EN**: Specifies if the core will have a clock enable pin (the equivalent of selecting the `CE` option in the CORE Generator GUI).
- **RST**: Specifies if the core will have a synchronous reset pin (the equivalent of selecting the `SCLR` option in the CORE Generator GUI).

System Generator automatically sets the Input Data Width parameter based on the signal properties of the `XN_RE` and `XN_IM` ports. System Generator supports only fixed-point data types and, hence, Data Format is not available as an option on the GUI.

Tab 3: Implementation

The Implementation tab is used to specify memory and optimization options in a similar way to page 3 of the CORE Generator GUI.

- **Number of stages using block RAM**: Specifies the number of stages for the Pipelined, Streaming I/O architecture that uses block RAM for data and phase factor storage. As dynamic list boxes are not offered with the System Generator GUI, this option displays the full range (0 to 11) selection, but allows the user to select only valid values as visible in the CORE Generator GUI.
- **FPGA Area Estimation**: See the System Generator documentation for detailed information about this option.

Bit-Accurate C Model

The FFT core has a bit-accurate C model designed for system modeling and selecting parameters before generating an FFT core. The model is bit-accurate but not cycle-accurate, so it produces exactly the same output data as the core on a frame-by-frame basis. However, it does not model the core latency or its interface signals.

The C model is generally required before generating an FFT core, so it is not delivered as an output of CORE Generator software. Instead it is available for download on the Xilinx LogiCORE IP FFT web page at www.xilinx.com/products/ipcenter/FFT.htm. The C model is available as a dynamically-linked library for 32-bit and 64-bit Windows platforms, and 32-bit and 64-bit Linux platforms. The C model is also available as a MATLAB software function for 32-bit Windows only. Download a zip file and unzip it to install the C model. A README.txt file describes the contents of the installed directory structure, and any further platform-specific installation instructions.

C Model Interface

The C model is used through three functions, declared in the header file `xfft_v7_1_bitacc_cmodel.h`:

```
struct xilinx_ip_xfft_v7_1_state*
xilinx_ip_xfft_v7_1_create_state(struct xilinx_ip_xfft_v7_1_generics generics);

int xilinx_ip_xfft_v7_1_bitacc_simulate
(
 struct xilinx_ip_xfft_v7_1_state*   state,
 struct xilinx_ip_xfft_v7_1_inputs   inputs,
 struct xilinx_ip_xfft_v7_1_outputs* outputs
 );

void xilinx_ip_xfft_v7_1_destroy_state(struct xilinx_ip_xfft_v7_1_state* state);
```

The first function, `xilinx_ip_xfft_v7_1_create_state`, creates a new state structure for the FFT C model, allocating memory to store the state as required, and returns a pointer to that state structure. The state structure contains all information required to define the FFT being modeled. The function is called with a structure containing the core generics: these are all of the parameters that define the bit-accurate numerical performance of the core, represented as integers, and are derived from the XCO parameters that are the result of selections in the CORE Generator GUI. The generics required for the C model and their mappings from XCO parameters are shown in Table 4.

Table 4: **C Model Generics**

Generic	Description	Range	XCO parameter and mapping
C_NFFT_MAX	$\log_2$(maximum point size)	3-16	transform_length: take $\log_2$
C_ARCH	Architecture	1-4	implementation_options: radix_4_burst_io => 1, radix_2_burst_io => 2, pipelined_streaming_io => 3, radix_2_lite_burst_io => 4
C_HAS_NFFT	Run-time configurable transform length	0,1	run_time_configurable_transform_length: false => 0, true => 1
C_INPUT_WIDTH	Input data width (bits)	8-34	input_width
C_TWIDDLE_WIDTH	Phase factor width (bits)	8-34	phase_factor_width

Table 4: **C Model Generics *(Cont'd)***

Generic	Description	Range	XCO parameter and mapping
C_USE_FLT_PT	Input/output data format	0,1	data_format: fixed_point => 0, floating_point => 1
C_HAS_SCALING	Scaling option: unscaled or not. Ignored when C_USE_FLT_PT = 1	0,1	scaling_options: unscaled => 0, scaled / block_floating_point => 1
C_HAS_BFP	Scaling option: if not unscaled, scaled or block floating-point. Ignored when C_USE_FLT_PT = 1	0,1	scaling_options: unscaled / scaled => 0, block_floating_point => 1
C_HAS_ROUNDING	Rounding mode. Ignored when C_USE_FLT_PT = 1	0,1	rounding_modes: truncation => 0, convergent_rounding => 1

After a state structure has been created, it can be used as many times as required to simulate the FFT core. A simulation is run using the second function, `xilinx_ip_xfft_v7_1_bitacc_simulate`. Call this function with the pointer to the existing state structure, and structures to hold the inputs and outputs of the C model. These input and output structures are fully defined and described in the C model header file. Note that memory for all input and output data arrays must be allocated by the calling program before simulating the C model.

Finally, the state structure must be destroyed to free up any memory used to store the state, using the third function, `xilinx_ip_xfft_v7_1_destroy_state`, called with the pointer to the existing state structure.

If the generics of the core need to be changed, destroy the existing state structure and create a new state structure using the new generics. There is no way to change the generics of an existing state structure.

An example C++ file, `run_bitacc_cmodel.c`, is included in the C model zip file. This shows all of the stages required to run the C model.

Due to differences between the FFT core and the C model in the order of operations within the processing phase, when using the Pipelined, Streaming I/O architecture, if fixed-point data is being processed, the scaling option is Scaled and overflow occurs, the `xk_re` and `xk_im` data outputs of the C model may not match the `XK_RE` and `XK_IM` data outputs of the core. The overflow output of the C model and the `OVFLO` output of the core (if present) do match in all cases. The overflow output of the C model is always set correctly when the scaling option is Scaled (when the C model generics `C_HAS_SCALING` = 1 and `C_HAS_BFP` = 0).

Therefore, Xilinx recommends that the overflow output of the C model is always checked when the scaling option is Scaled and the architecture is Pipelined, Streaming I/O, and if overflow has occurred (overflow output = 1), the `xk_re` and `xk_im` outputs of the C model are ignored. This is the only case where the C model is not entirely bit-accurate to the core.

Using the C Model to Select a Scaling Schedule

When the scaling option for the FFT core is Scaled, the user is given great flexibility to set the scaling schedule that determines by how much to scale data values at each stage of the FFT processing phase. See "Forward/Inverse and Scaling Schedule," page 26. It can be difficult to choose the best scaling schedule that avoids overflow in a sufficiently large proportion of frames for a particular type of input data. The C model is a tool that can help with the selection of a scaling schedule. A process for this is as follows:

1. Create a set of frames of typical FFT input data for the intended application.
2. Create a state structure using the required generics. Set the scaling option to Scaled by setting the C model generics `C_HAS_SCALING` = 1 and `C_HAS_BFP` = 0.
3. Set the scaling schedule in the structure of inputs to some initial scaling schedule, such as the reset value of 1 in each stage for Radix-2, Burst I/O and Radix-2 Lite, Burst I/O architectures, or 2 in each stage for Radix-4, Burst I/O, and Pipelined, Streaming I/O architectures.
4. Simulate the C model with each frame of typical input data in turn. Count the number of frames in which overflow occurred (overflow output was 1).
5. If the percentage of frames in which overflow occurred is lower than the acceptable overflow rate, reduce the scaling value in one or more stages in the scaling schedule. If the percentage of frames in which overflow occurred is higher than the acceptable overflow rate, increase the scaling value in one or more stages in the scaling schedule.
6. Repeat stages 4 and 5 until the percentage of frames in which overflow occurred matches the acceptable overflow rate.

This process produces a scaling schedule that is tailored to the typical FFT input data for the intended application.

Control Signals and Timing

Clock Enable

If the Clock Enable pin is present on the core, driving the pin Low will pause the core in its current state. All logic within the core will be paused. Driving the `CE` pin High will allow the core to continue processing.

Synchronous Clear

Synchronous Clear overrides Clock Enable if both are present on the core. Asserting the Synchronous Clear (`SCLR`) pin results in all output pins, internal counters, and state variables being reset to their initial values. All pending load processes, transform calculations, and unload processes stop and are reinitialized. `NFFT` is set to the largest FFT point size permitted (the Transform Length value set in the GUI). The scaling schedule is set to *I/N*. For the Radix-4, Burst I/O and Pipelined, Streaming I/O architectures with a non-power-of-four point size, the last stage has a scaling of 1, and the rest have a scaling of 2. See Table 5.

Table 5: **Synchronous Clear Reset Values**

Signal	Initial / Reset Value
NFFT	maximum point size = *N*
FWD_INV	Forward = 1
SCALE_SCH	1/*N* [10 10... 10] for Radix-4, Burst I/O or Pipelined, Streaming I/O architectures when *N* is a power of 4. [01 10... 10] for Radix-4, Burst I/O or Pipelined, Streaming I/O architectures when *N* is not a power of 4. [01 01... 01] for Radix-2, Burst I/O or Radix-2 Lite, Burst I/O architectures

Note: If the run-time configurable transform length option is selected, asserting the NFFT_WE pin resets the core in the same way as asserting the SCLR pin, except that NFFT_WE does not reset the latched scaling schedule and transform type (forward or inverse). Note that NFFT_WE *does not* override Clock Enable, unlike Synchronous Clear. Therefore, Synchronous Clear may not be required in addition to run-time configurable transform length. Omitting Synchronous Clear may result in a saving of logic resources and may allow a higher maximum clock frequency.

Transform Size

The transform point size can be set through the NFFT port if the run-time configurable transform length option is selected. Valid settings and the corresponding transform sizes are provided in Table 6. If the NFFT value entered is too large, the core sets itself to the largest available point size (selected in the GUI). If the value is too small, the core sets itself to the smallest available point size: 64 for the Radix-4, Burst I/O architecture and 8 for the other architectures.

NFFT values are read in on the rising clock edge when NFFT_WE is High. A new transform size re-times all current processes within the core, so every time a transform size is latched in, regardless of whether or not the new point size differs from the current point size, the core is internally reset. (FWD_INV and SCALE_SCH are not reset.) Holding NFFT_WE High continues to reset the core on every clock cycle.

Table 6: **Valid NFFT Settings**

NFFT[4:0]	Transform size (*N*)
00011	8
00100	16
00101	32
00110	64
00111	128
01000	256
01001	512
01010	1024
01011	2048
01100	4096
01101	8192
01110	16384
01111	32768
10000	65536

Forward/Inverse and Scaling Schedule

The transform type (forward or inverse) and the scaling schedule can be set frame-by-frame without interrupting frame processing. Both the transform type and the scaling schedule can be set independently for each channel in a multichannel core. A single channel core uses the FWD_INV pin to set the transform type and the SCALE_SCH bus to set the scaling schedule. A multichannel core has a FWD_INV pin for each channel, named FWD_INV0, FWD_INV1, ... and a SCALE_SCH bus for each channel, named SCALE_SCH0, SCALE_SCH1, ...

The transform type can be set using the FWD_INV pin. Setting FWD_INV to 0 produces an inverse FFT, and setting FWD_INV to 1 creates the forward transform.

Burst I/O Architectures

The scaling performed during successive stages can be set via the SCALE_SCH bus. For the Radix-4, Burst I/O and Radix-2 architectures, the value of the SCALE_SCH bus is used as pairs of bits [... N4, N3, N2, N1, N0], each pair representing the scaling value for the corresponding stage. Stages are computed starting with stage 0 as the two LSBs. There are $\log_4$(point size) stages for Radix-4 and $\log_2$(point size) stages for Radix-2. In each stage, the data can be shifted by 0, 1, 2, or 3 bits, which corresponds to SCALE_SCH values of 00, 01, 10, and 11. For example, for Radix-4, when N = 1024, [01 10 00 11 10] translates to a right shift by 2 for stage 0, shift by 3 for stage 1, no shift for stage 3, a shift of 2 in stage 3, and a shift of 1 for stage 4 (there are $\log_4(1024) = 5$ Radix-4 stages). This scaling schedule scales by a total of 8 bits which gives a scaling factor of 1/256. The conservative schedule SCALE_SCH = [10 10 10 10 11] completely avoids overflows in the Radix-4, Burst I/O architecture. For the Radix-2, Burst I/O and Radix-2 Lite, Burst I/O architectures, the conservative scaling schedule of [01 01 01 01 01 01 01 01 01 10] prevents overflow for N = 1024 (there are $\log_2(1024) = 10$ Radix-2 stages).

Pipelined, Streaming I/O Architecture

For the Pipelined, Streaming I/O architecture, consider every pair of adjacent Radix-2 stages as a group. That is, group 0 contains stage 0 and 1, group 1 contains stage 2 and 3, and so forth. The value of the SCALE_SCH bus is also used as pairs of bits [... N4, N3, N2, N1, N0]. Each pair represents the scaling value for the corresponding group of two stages. Groups are computed starting with group 0 as the two LSBs. In each group, the data can be shifted by 0, 1, 2, or 3 bits which corresponds to SCALE_SCH values of 00, 01, 10, and 11. For example, when N = 1024, [10 10 00 01 11] translates to a right shift by 3 for group 0 (stages 0 and 1), shift by 1 for group 1 (stages 2 and 3), no shift for group 3 (stages 4 and 5), a shift of 2 in group 3 (stages 6 and 7), and a shift of 2 for group 4 (stages 8 and 9). The conservative schedule SCALE_SCH = [10 10 10 10 11] completely avoids overflows in the Pipelined, Streaming I/O architecture. When the point size is not a power of 4, the last group only contains one stage, and the maximum bit growth for the last group is one bit. Therefore, the two MSBs of the scaling schedule can only be 00 or 01. A conservative scaling schedule for N = 512 is SCALE_SCH = [01 10 10 10 11].

The user is allowed great flexibility to set the transform type (Forward/Inverse) and the scaling schedule. The FWD_INV and SCALE_SCH values are latched into temporary registers whenever the corresponding WE pins are High. FWD_INV_WE and SCALE_SCH_WE can be asserted at any time until 3 cycles after START is asserted, irrespective of the Input Data Timing parameter value. The core then reads these temporary registers, and these are the values that are used for that frame of data. There is no way to alter those values once the transform calculation phase has started. Any WE assertions later than 3 cycles after START is asserted affect the frame that follows.

For a multichannel core, there are separate FWD_INV_WE and SCALE_SCH_WE pins for each channel, named FWD_INV0_WE, FWD_INV1_WE, ... and SCALE_SCH0_WE, SCALE_SCH1_WE, ...

Both the scaling schedule and the transform type are registered internally, so there is no need to hold these values on the pins. Also, if the scaling and transform type are constant through multiple frames (that is, no new values are latched in), registered values apply for successive frames. The scaling schedule and transform type are *not* reset when NFFT_WE is asserted.

The initial value and reset value of FWD_INV is forward = 1. The scaling schedule is set to *I/N*. That translates to [10 10 10 10... 10] for the Radix-4, Burst I/O and Pipelined, Streaming I/O architectures, and [01 01... 01] for the Radix-2 architectures. The core uses the (2*number of stages) LSBs for the scaling schedule. So, when the point size decreases, the leftover MSBs are ignored. However, all bits are latched into the core on SCALE_SCH_WE and are used in later transforms if the point size increases.

Cyclic Prefix Insertion

Cyclic prefix insertion takes a section of the output of the FFT and prefixes it to the beginning of the transform. The resultant output data consists of the cyclic prefix (a copy of the end of the output data) followed by the complete output data, all in natural order. Cyclic prefix insertion is only available when output ordering is Natural Order.

When cyclic prefix insertion is used, the length of the cyclic prefix can be set frame-by-frame without interrupting frame processing. The cyclic prefix length can be any number of samples from zero to one less than the point size. The cyclic prefix length is set by the CP_LEN bus. For example, when $N = 1024$, the cyclic prefix length can be from 0 to 1023 samples, and a CP_LEN value of 0010010110 will produce a cyclic prefix consisting of the last 150 samples of the output data.

The user is allowed great flexibility to set the cyclic prefix length. The CP_LEN value is latched into a temporary register whenever the CP_LEN_WE pin is High. CP_LEN_WE can be asserted at any time before the frame of data is loaded in. The core reads this temporary register three cycles after START is asserted, irrespective of the Input Data Timing parameter. This is the value that is used for the current frame of data. There is no way to alter this value once the transform calculation phase has started. Any CP_LEN_WE assertions later than three cycles after START is asserted affect the frame that follows.

The cyclic prefix length is registered internally, so there is no need to hold the value on the CP_LEN bus. Also, if the cyclic prefix length is constant through multiple frames (that is, no new values are latched in), registered values apply for successive frames. The cyclic prefix length is not reset when NFFT_WE is asserted.

The initial value and reset value of CP_LEN is 0 (no cyclic prefix). The core uses the $\log_2$(point size) MSBs of CP_LEN for the cyclic prefix length. So, when the point size decreases, the leftover LSBs are ignored. This effectively scales the cyclic prefix length with the point size, keeping them in approximately constant proportion. However, all bits of CP_LEN are latched into the core on CP_LEN_WE and are used in later transforms if the point size increases.

Overflow

Fixed-Point Data

The Overflow (OVFLO) signal is only available when the Scaled arithmetic is used. OVFLO is driven High during unloading if any point in the data frame overflowed. For a multichannel core, there is a separate OVFLO output for each channel, named OVFLO0, OVFLO1, ... For the Burst I/O architectures, the OVFLO signal goes High as soon as an overflow occurs during the computation and remain High during the entire time the frame is unloading. For the Pipelined, Streaming I/O architecture, the OVFLO signal goes High during unloading as soon as an overflow is detected in that frame and is held high for the remainder of the frame.

When an overflow occurs in the core, the data is wrapped rather than saturated, resulting in the transformed data becoming unusable for most applications.

Floating-Point Data

The Overflow signal is used to indicate an exponent overflow when the FFT is processing floating-point data. When an exponent overflow occurs, the OVFLO signal goes High as soon as an overflow is detected in that frame, and remains High for the remainder of the frame. This behavior is the same for both Burst I/O and Pipelined, Streaming I/O architectures, which is different from the Overflow behavior for fixed-point data described previously.

The output sample which overflowed will be set to +/- Infinity, depending on the sign of the internal result.

The Overflow signal will not be asserted when a NaN value is present on the output. NaN values can only occur at the FFT output when the input data frame contains NaN or +/- Infinity samples.

Block Exponent

The Block Exponent (BLK_EXP) signal (used only with the block floating-point option) contains the block exponent. For a multichannel core, there is a separate BLK_EXP output for each channel, named BLK_EXP0, BLK_EXP1, ... The signal is valid during the unloading of the data frame. The value present on the port represents the total number of bits the data was scaled during the transform. For example, if BLK_EXP has a value of 00101 = 5, this means the output data (XK_RE, XK_IM) was scaled by 5 bits (shifted right by 5 bits), or in other words, was divided by 32, to fully utilize the available dynamic range of the output data path without overflowing.

Timing for the Pipelined, Streaming I/O Architecture

Setting Up and Starting the Transform

Asserting START starts the data loading phase, which immediately flows into the transform calculation phase and then the data unloading phase. Pulsing START once allows the transform calculation for a single frame. Pulsing START every N clock cycles allows continuous data processing. Alternatively, holding START High also allows continuous data processing (see Figure 8 or Figure 9 if cyclic prefix insertion is used). START is ignored except when the core can begin loading a new frame, that is, when no data is being loaded, or the last value in the data frame is being loaded. If no NFFT_WE, FWD_INV_WE, or SCALE_SCH_WE were asserted before the initial START, then the defaults are used. This architecture can also support extended intervals between frames (Figure 10). Simply assert START at any time to begin data loading. After the data frame is loaded, the core proceeds to calculate the

transform and then output the results. Figure 10 shows the timing of entire frames. It does not show the small skews between signals which occur at the start and end of frames.

Applying Data

Data is applied in a contiguous burst. The point at which data input should start relative to the START pulse is determined by the Input Data Timing parameter set in the GUI.

If "No offset" was selected for the Input Data Timing parameter, the input data (XN_RE, XN_IM) corresponding to the given XN_INDEX should arrive on the same cycle as the XN_INDEX it matches. The first data sample should therefore be applied as soon as RFD goes High, such that the first sample pair is read into the core on the first transition of XN_INDEX.

If "3 clock cycle offset" was selected for the Input Data Timing parameter, the input data (XN_RE, XN_IM) corresponding to the given XN_INDEX should arrive three clock cycles later than the XN_INDEX it matches (see Figure 11). In this way, XN_INDEX can be used to address external memory or a frame buffer storing the input data.

RFD remains High with XN_INDEX during the loading phase and so indicates that data may be input.

Data Processing and Data Output

BUSY goes High while the core is calculating the transform. DONE goes High when calculation is complete. EDONE goes High one cycle before that, that is, during the last cycle of the calculation phase. The cycle in which DONE goes High, the core begins unloading. During the unloading phase, while valid output results are present on XK_RE/XK_IM, DV (Data Valid) is High. During unloading, XK_INDEX corresponds to the XK_RE/XK_IM being presented. If cyclic prefix insertion is used, the cyclic prefix is unloaded first. CPV goes High to indicate that the cyclic prefix is being unloaded, and XK_INDEX counts from (point size) - (cyclic prefix length) up to (point size) - 1. After the cyclic prefix has been unloaded, or if the cyclic prefix length is zero, or if cyclic prefix insertion is not used, the whole frame of output data is unloaded. CPV goes Low (if present) and XK_INDEX counts from 0 up to (point size) - 1.

Cyclic Prefix Considerations

If cyclic prefix insertion is used, more samples are unloaded from the core than are loaded. Therefore, the core cannot continuously stream frames, but must insert a gap of (cyclic prefix length) clock cycles in between each frame of input data to accommodate the additional clock cycles required to unload the cyclic prefix. This is indicated by the Ready For Start (RFS) pin. RFS goes High when the core is ready for the START pin to be asserted to begin loading the next frame of data. START is ignored except when RFS is High. RFS remains low for (cyclic prefix length) clock cycles after RFD has gone Low, to allow for unloading the cyclic prefix.

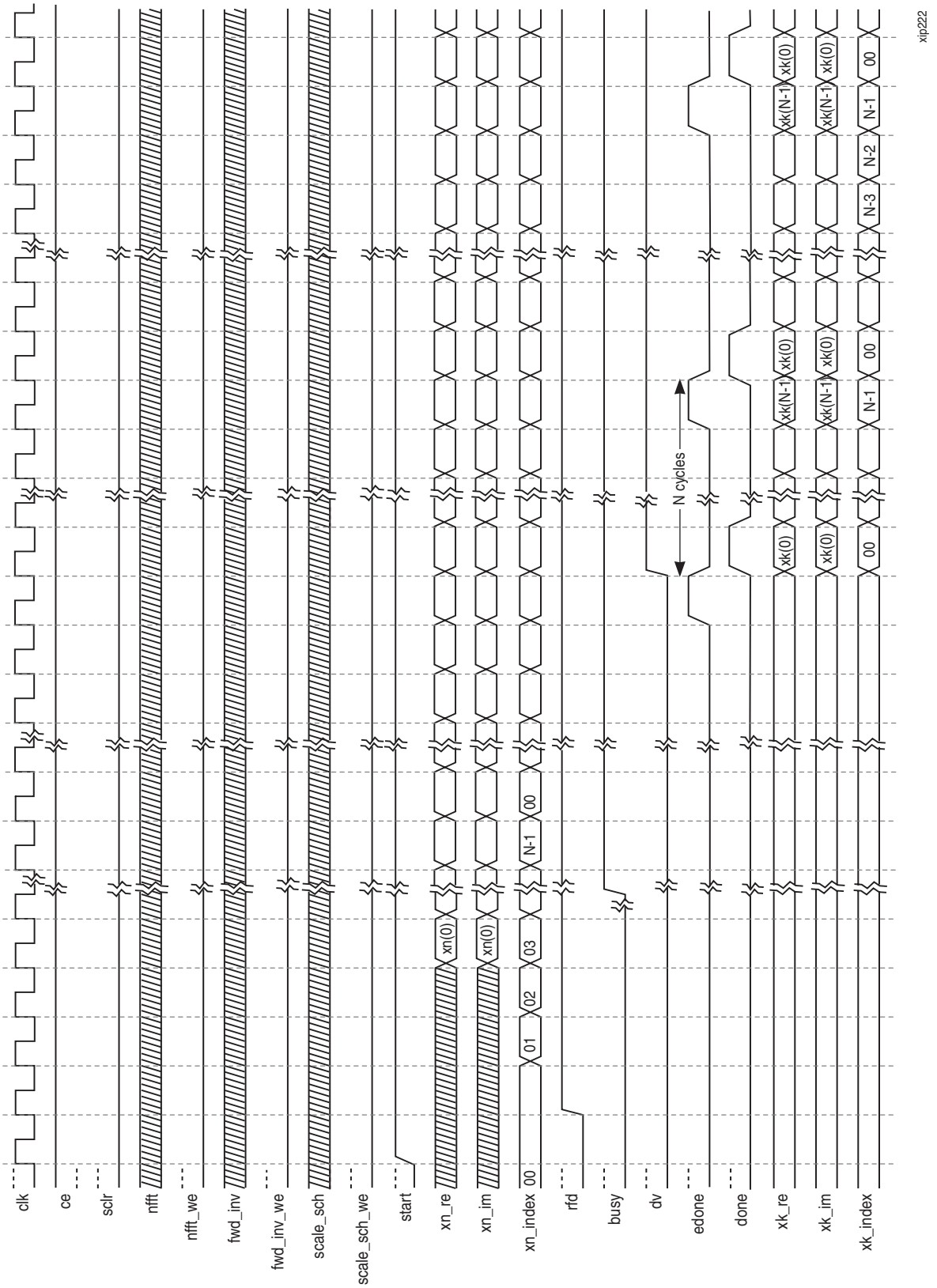


Figure 8: **Timing for Continuous Streaming Data**

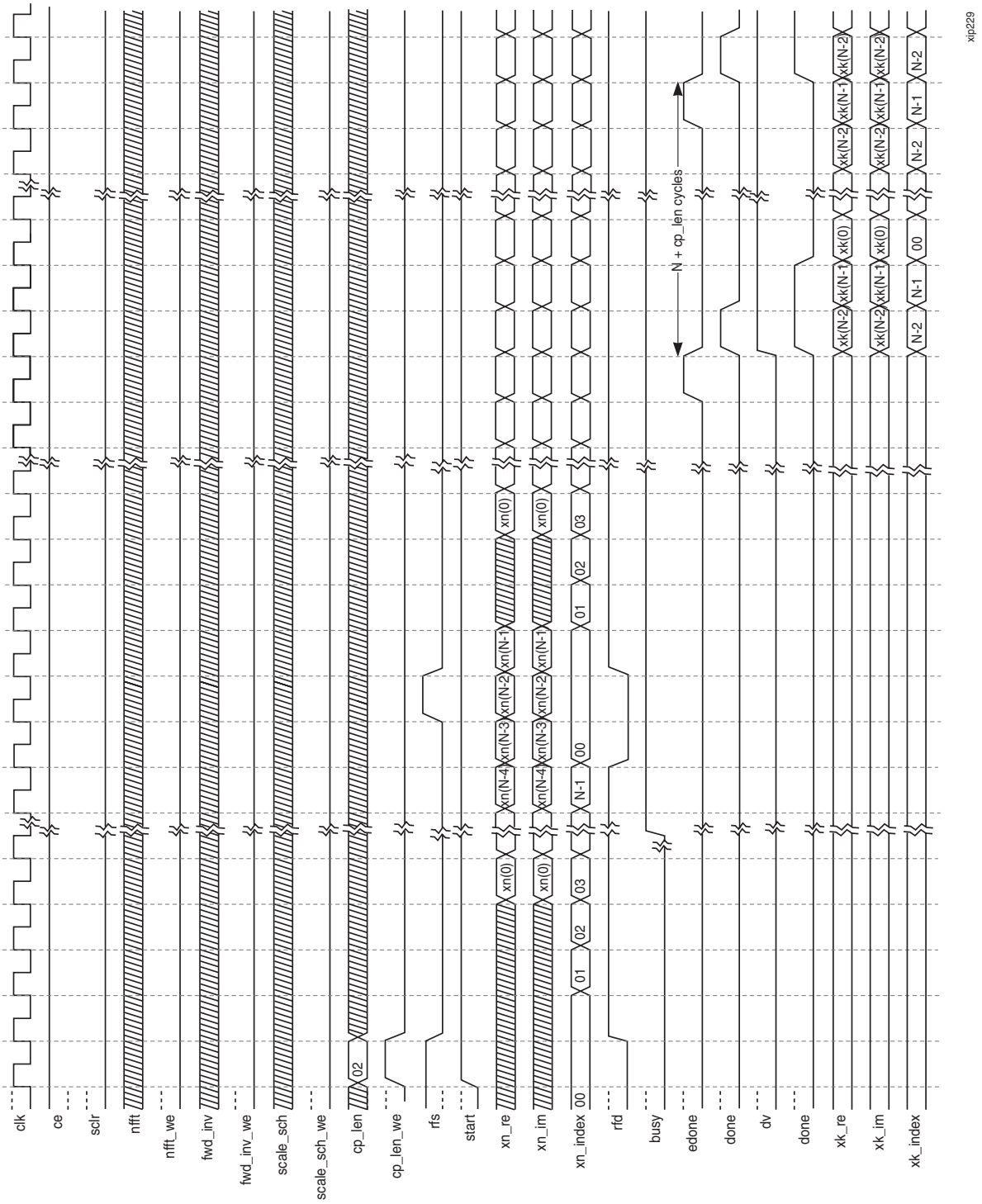
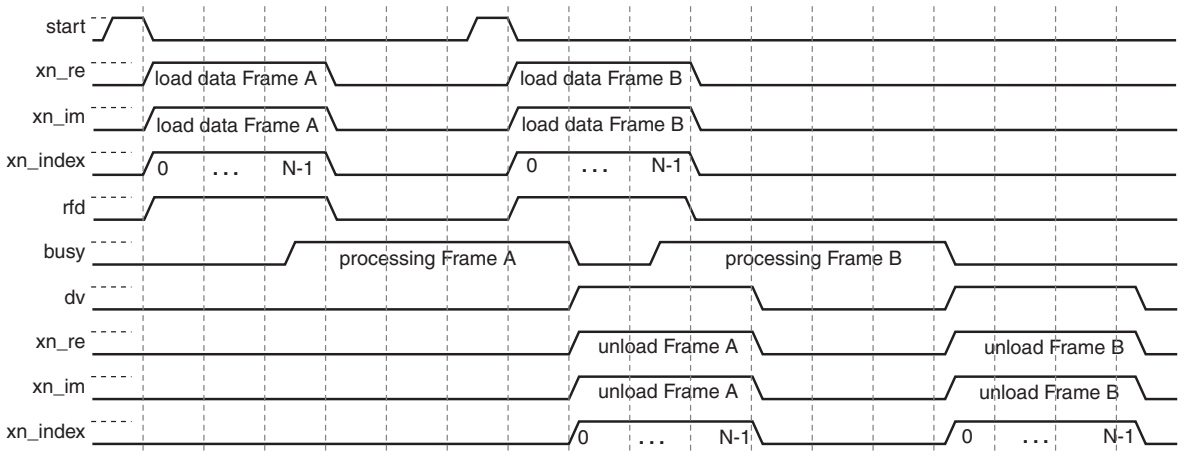


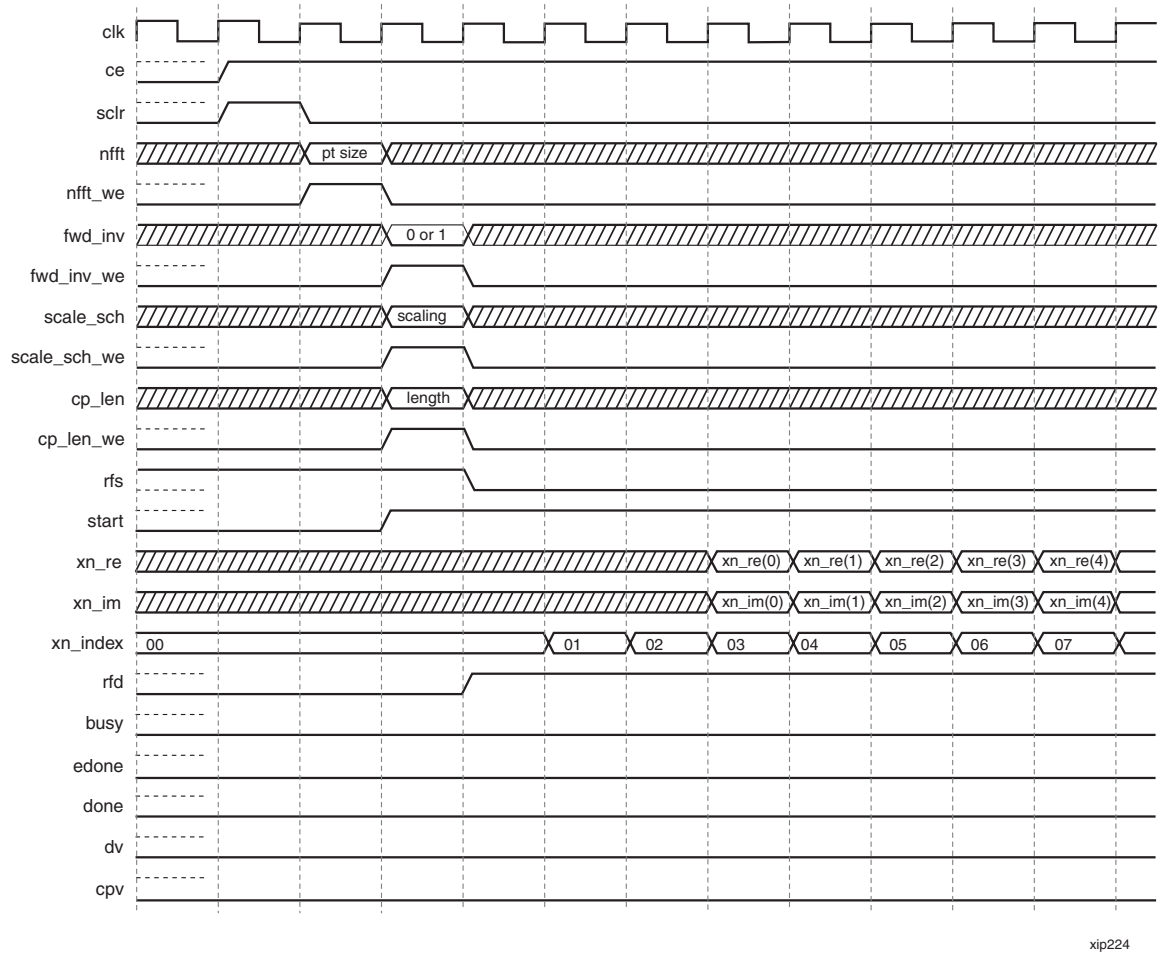
Figure 9: **Timing for Continuous Streaming Data with Cyclic Prefix Insertion of Length 2**



Note:
All transitions are synchronous with the rising edge of the clock.

xip223

Figure 10: **Timing for Non-Continuous Data Stream**



xip224

Figure 11: **Beginning of Data Frame (Input Data Timing = "3 clock cycle offset")**

Timing for the Radix-4, Burst I/O, Radix-2, Burst I/O, and Radix-2 Lite, Burst I/O Architectures

Setting Up and Starting the Transform

The START signal begins the data loading phase, which leads directly to the calculation phase. START is only valid when the core is idle or if the core is in bit reversed output mode and unloading the processed data.

Applying Data

Data is applied in a contiguous burst. The point at which data input should start relative to the START pulse is determined by the Input Data Timing parameter set in the GUI.

If "No offset" was selected for the Input Data Timing parameter, the input data (XN_RE, XN_IM) corresponding to the given XN_INDEX should arrive on the same cycle as the XN_INDEX it matches. The first data sample should therefore be applied as soon as RFD goes High, such that the first sample pair is read into the core on the first transition of XN_INDEX.

If "3 clock cycle offset" was selected for the Input Data Timing parameter, the input data (XN_RE, XN_IM) corresponding to the given XN_INDEX should arrive three clock cycles later than the XN_INDEX it matches (see Figure 11). In this way, XN_INDEX can be used to address external memory or a frame buffer storing the input data.

RFD remains High with XN_INDEX during the loading phase and so indicates that data may be input.

Data Processing

BUSY goes High while the core is calculating the transform. DONE goes High when calculation is complete. EDONE goes High one cycle before that, that is, during the last cycle of the calculation phase.

Data Output

After the data is loaded and processed, two options are available to unload data:

- If Natural Order output order was selected, UNLOAD should be asserted (Figure 12, or Figure 13 if cyclic prefix insertion is used) to output the data. During the unloading phase, while valid output results are present on XK_RE/XK_IM, DV (Data Valid) is High. During unloading, XK_INDEX corresponds to the XK_RE/XK_IM being presented. If cyclic prefix insertion is used, the cyclic prefix is unloaded first. CPV goes High to indicate that the cyclic prefix is being unloaded, and XK_INDEX counts from (point size) - (cyclic prefix length) up to (point size) - 1. After the cyclic prefix has been unloaded, or if the cyclic prefix length is zero, or if cyclic prefix insertion is not used, the whole frame of output data is unloaded. CPV goes Low (if present) and XK_INDEX counts from 0 up to (point size) - 1. UNLOAD can be asserted any time from when EDONE goes High. UNLOAD is ignored except when the core can begin unloading. In addition to using pulses, START and UNLOAD can be tied High (Figure 14). In this case, the core continuously loads, processes, and unloads data. Figure 14 shows the timing of entire frames. It does not show the small skews between signals which occur at the start and end of frames and does not show the length of each phase of the transform to scale. The processing time may be much longer than the time required to input or output a frame.
- If Bit/Digit-Reversed output order was selected, the user can assert START again (Figure 15). While the next frame of data is loaded, the results are output at the same time. START can be asserted any time from when EDONE goes High. If START is tied High, the core continuously loads/unloads then processes, loads/unloads then processes, and so on (Figure 16).

DV remains High during data unloading in both cases.

There is a latency of several clock cycles after triggering an unload with UNLOAD or START before the output data XK_RE/XK_IM is presented. This latency varies as a function of several core parameters, but the output data is qualified by DV (Data Valid) and XK_INDEX, so should be considered as a handshake.

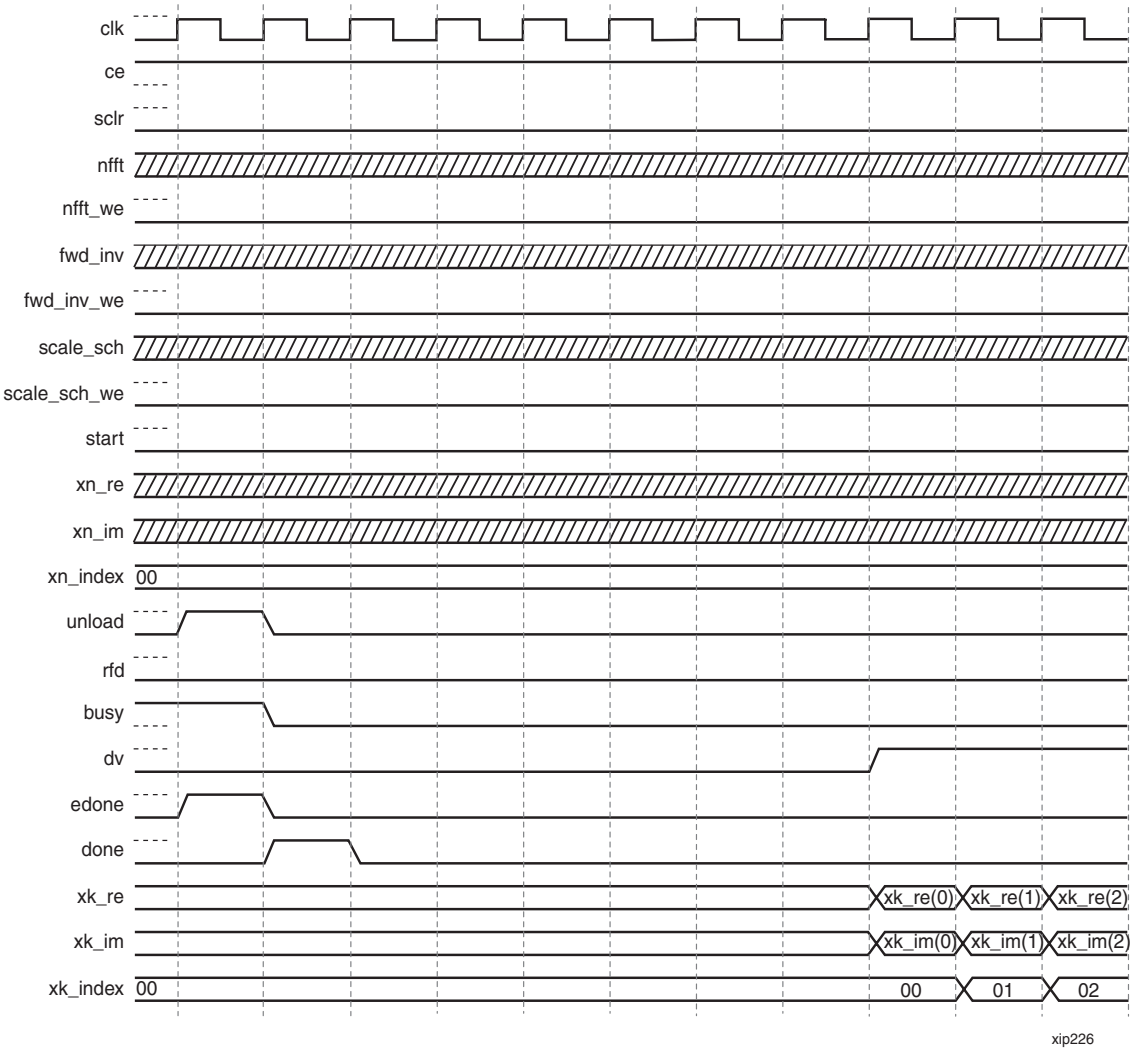


Figure 12: **Unload Output Results in Natural Order**

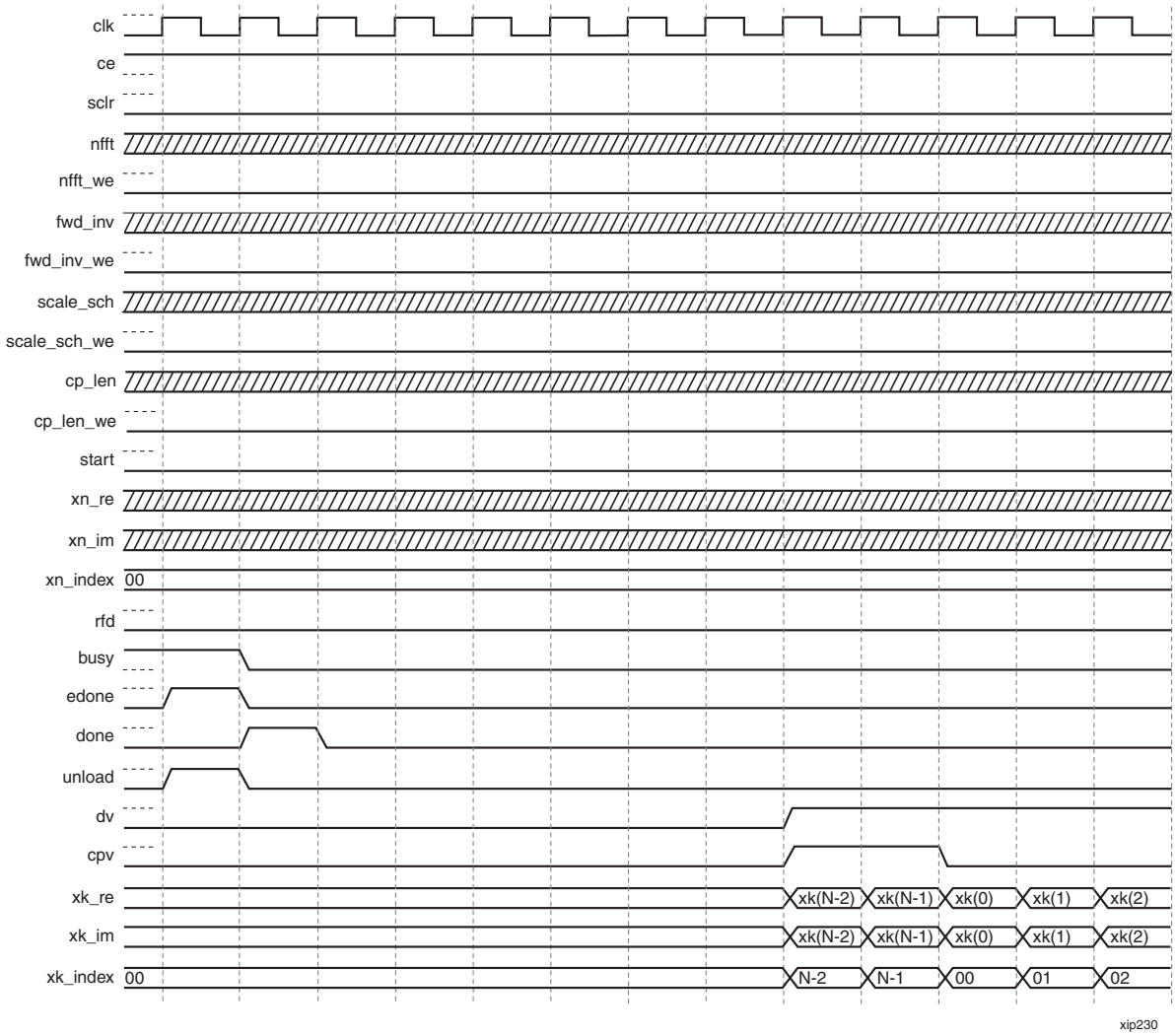


Figure 13: **Unload Output Results in Natural Order with Cyclic Prefix Insertion of Length 2**

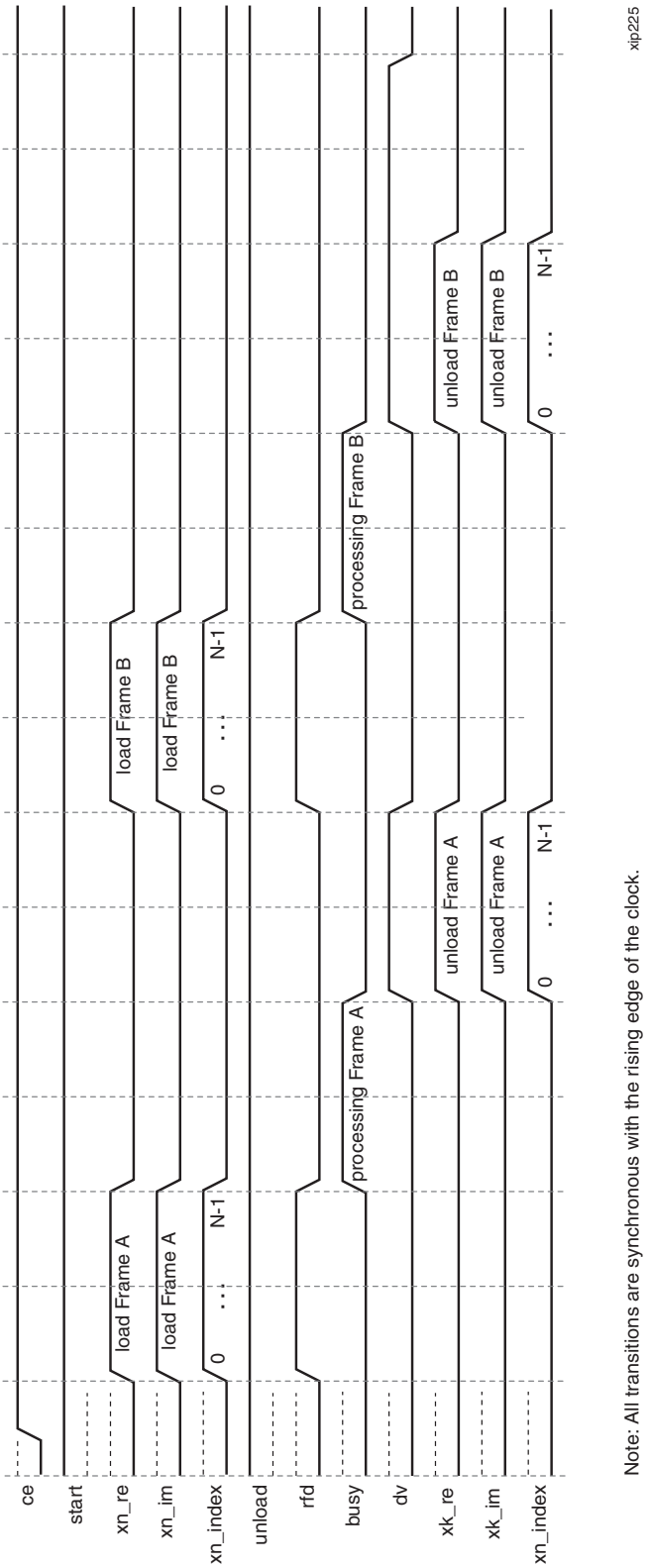


Figure 14: **Timing for Burst I/O Solutions with Natural Order Output**

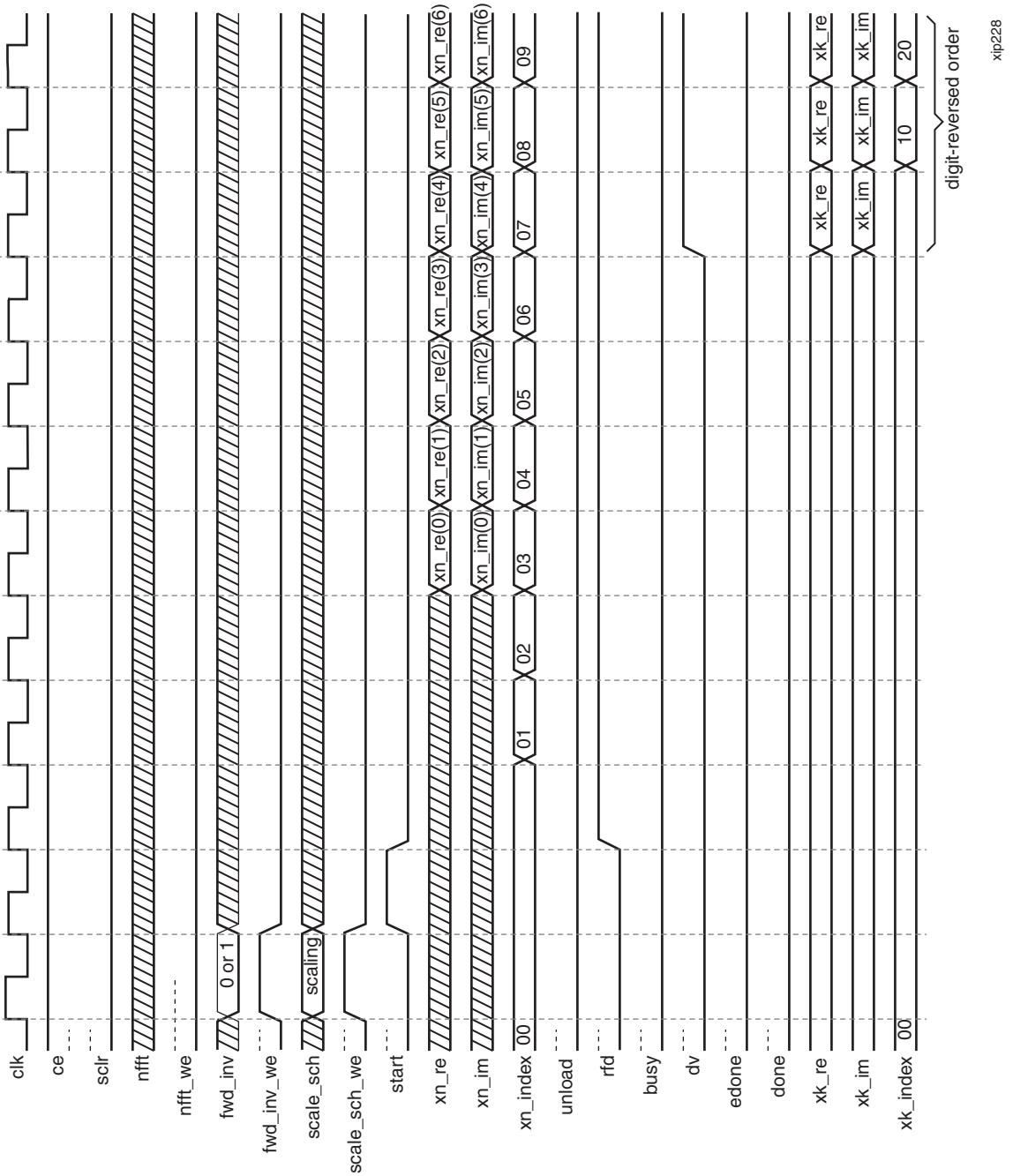


Figure 15: **Unload Results in Bit/Digit Reversed Order (Input Data Timing = "3 clock cycle offset")**

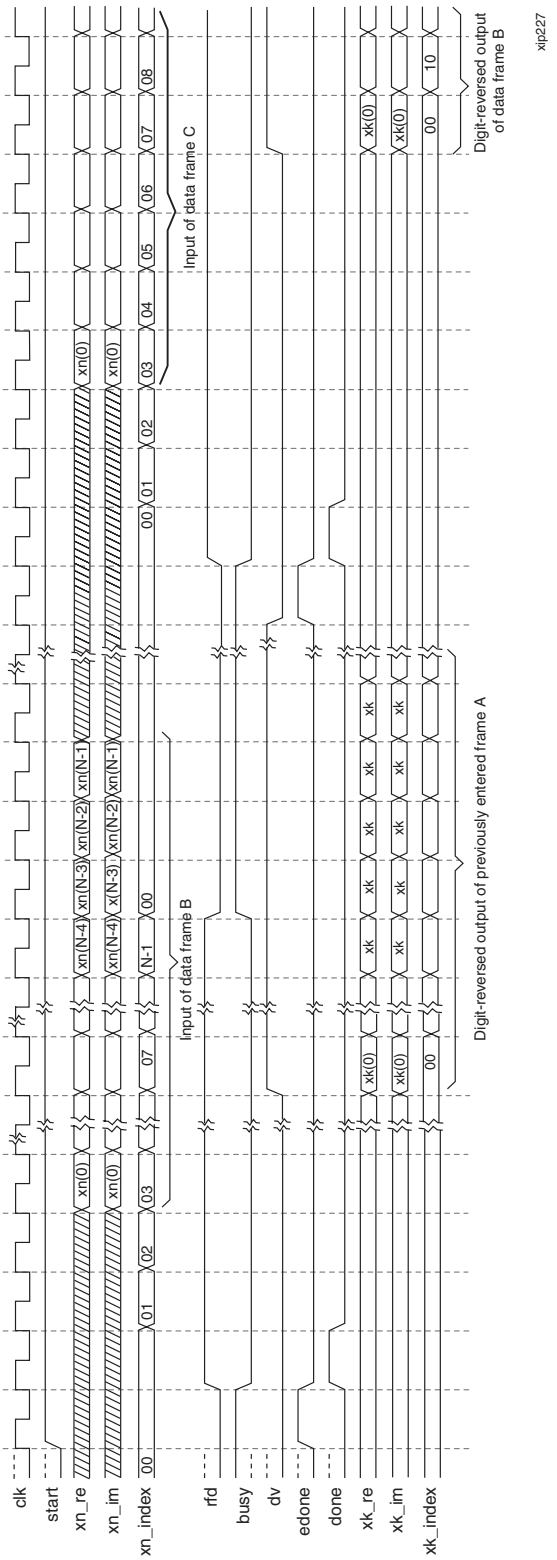


Figure 16: **Continuous Processing with Bit/Digit Reversed Order (Input Data Timing = "3 clock cycle offset")**

Known Device-Specific Constraints

This section details issues which may be encountered when mapping an FFT core to a particular device. In many cases, it is possible to work around these issues by adjusting the core configuration without having to alter the target FPGA device.

Spartan-3 FPGA Constraints

Explanation

In these device architectures, the multiplier site adjacent to the location of a 512x36 block RAM component must remain free because of interconnect resource sharing between the multipliers and block RAMs. This means that the adjacent block RAM can be used only up to 18 bits wide when the multiplier is used.

Error Message

During Place and Route, the Place tool generates a message similar to the following:

```
ERROR:Place:341 - The design contains 6 Block RAM components that are configured
as 512x36 Block RAMs and 168 Multiplier components. The Multiplier site adjacent
to the location of a 512x36 Block RAM component must remain free because of
resource sharing. Therefore a device must have at least 174 Multiplier sites for
this design to fit. The current device has only 168 Multiplier sites.
```

Placer errors 419 and 665 may also be present.

Solution

There are a number of solutions for this issue:

- Reduce the FFT input data width and/or phase factor width to 18 bits or 17 bits, respectively, to allow adjacent block RAMs and multipliers to be used
- If the FFT uses the Pipelined, Streaming I/O architecture, reduce the value of the *Number Of Stages Using Block RAM* parameter to reduce the number of block RAMs required. This would increase the number of slices used by the core.
- If the FFT uses a Burst I/O architecture, use distributed RAM for the data or phase factor memory, or use the hybrid memory optimization (if available) to reduce the number of block RAMs required. This would increase the number of slices used by the core.
- If an unscaled FFT is to be implemented, utilize a scaled or block floating-point FFT instead to reduce the output width to 18 bits or less to allow adjacent block RAMs and multipliers to be used.
- Use a larger device with more block RAM and multiplier components.

Spartan-3A DSP FPGA Constraints

Explanation

The Spartan-3A DSP device has a split in the left-most and right-most XtremeDSP slice columns to accommodate the clock tiles. The complex multipliers in the FFT core use the dedicated cascade routing between the XtremeDSP slices to enable high performance and reduce power consumption. The cascade routing cannot cross the clock tile in these particular columns.

In densely-packed devices where many XtremeDSP slices have been used, the placer may have no option but to attempt to place the cascaded XtremeDSP slices in these split columns, which is not possible. This can occur in multichannel Burst I/O FFTs and large Pipelined, Streaming I/O FFTs.

Error Message

During Place and Route, the Place tool generates a message similar to the following:

```
WARNING:Place:119 - Unable to find location. DSP48A component
blk00000003/blk00002ec4 not placed.
WARNING:Place:119 - Unable to find location. DSP48A component
blk00000003/blk00002ec9 not placed.
WARNING:Place:119 - Unable to find location. DSP48A component
blk00000003/blk00002ec8 not placed.
Comps belong to structure: Multiplier Cascade RPM
<a number of instance names are listed>
ERROR:Place:120 - There were not enough sites to place all selected components.
```

Solutions

- If the FFT option to optimize complex multipliers for speed using XtremeDSP slices has been checked, un-checking this option will use fewer XtremeDSP slices, and may permit packing in the device. This may impact the maximum achievable clock frequency.
- If the option to optimize butterflies using XtremeDSP slices has been checked, un-checking this option will free XtremeDSP slice locations which may allow placement to succeed. This may impact the maximum achievable clock frequency.
- Reduce the data and phase factor widths until the number of XtremeDSP slices is reduced. Because the phase factor width is increased by 1 bit internally, reducing it to 17 bits or less will allow a smaller complex multiplier architecture to be utilized. This will not impact the maximum achievable clock frequency (and may improve it), but yields a small reduction in data precision.
- Use a larger device. The left-most and right-most columns on the XC3SD1800A device are shorter (10 XtremeDSP slices) than the equivalent columns on the XC3SD3400A device (12 XtremeDSP slices). See [Ref 5] for further details on Spartan-3A DSP XtremeDSP slices.

Performance and Resource Usage

The following tables list the resource usage and transform time for a selected set of parameters. This core does not use placement constraints, allowing Place and Route full flexibility. The slice count, block RAM count, and XtremeDSP slice count are listed. The maximum clock frequency is listed with the transform latency. The latency is from asserting the `START` input to the last sample of output data coming out of the core, assuming that the `UNLOAD` input is asserted as soon as possible if present. The following device architectures are represented:

- "Virtex-6 FPGA Family"
- "Virtex-5 FPGA Family"
- "Spartan-6 Family"
- "Spartan-3A DSP Family"

The maximum clock frequency for each test was determined iteratively. For the determination of maximum frequency, the core was generated with double registers on each input and output. The registers directly connected to the core run on the core clock, whereas the outer registers run off a separate clock. This ensures that all paths in the core are included in the timing constraint without artificially distorting the design to fit the chip. The slowest speed grade is used for each family. The parameters used for map and par are as follows:

```
map -pr b -ol high
par -ol high
```

The maximum achievable clock frequency and the resource counts may also be affected by other tools options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors.

Improved performance or resource usage may be achieved by applying an area group, or using map arguments such as "-lc area." Consult the ISE 12.1 software documentation for more details on available options.

When comparing performance and resource usage with Fast Fourier Transform v5.0, note that the -c 1 map option was not used in the preceding arguments, leading to higher slice counts, but improved performance.

Virtex-6 FPGA Family

Table 7 shows performance and resource usage numbers for Virtex-6 FPGAs. A range of FFT cores is shown for several typical applications: Baseband 3GPP LTE, Baseband OFDM, CT scanners, Ultrasound, Test and measurement, and Radar. The parameters for each core are shown in the table. None of the optional pins (CE, SCLR, OVFLO) are used. Hybrid RAM is not used. The performance and resource usage numbers were produced using ISE 12.1 software, with speed file version "ADVANCED 1.05 2010-03-16."

Table 7: **Virtex-6 FPGA Family Performance and Resource Utilization**

Application	Channels	Point Size	Implementation(1)	Variable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	LUT/FF Pairs	LUTs	FFs	18k Block RAMs(8)	XtremeDSP Slices	Max Clock Frequency(9)	Latency (cycles)(10)	Latency (µs)(11)
Baseband 3GPP LTE	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX75T	818	802	1031	3	2	395	12453	31.53
	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC6VLX75T	803	785	1033	3	3	395	12473	31.58
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX75T	911	883	1084	5	2	395	26804	67.86
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC6VLX75T	898	866	1086	5	3	395	26826	67.91
	4	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	1828	1798	2603	9	8	395	12453	31.53
	8	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX240T	3191	3107	4699	17	16	366	12453	34.02
	4	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	1926	1889	2674	17	8	395	26804	67.86
	8	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6VLX240T	4089	2461	4794	33	16	360	26804	74.46
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX75T	1023	1002	1204	3	3	395	7364	18.64
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC6VLX75T	893	868	1090	3	6	395	7354	18.62
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX75T	1112	1092	1265	5	3	395	15575	39.43
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC6VLX75T	953	930	1153	5	6	395	15564	39.40
	2	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	1604	1576	1978	5	6	395	7364	18.64
	4	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	2772	2744	3526	9	12	395	7364	18.64
	8	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX240T	5100	5045	6622	17	24	366	7364	20.12
	2	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	1735	1715	2077	9	6	395	15575	39.43
	4	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX130T	2991	2952	3701	17	12	374	15575	41.64
	8	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6VLX240T	6201	4832	6949	33	24	339	15575	45.94
OFDM	1	256	R2	Y	12	12	S	T	N	Y	D	-	N	XC6VLX75T	867	850	817	0	3	450	1652	3.67
	1	256	R2	Y	12	12	S	T	N	Y	B	-	N	XC6VLX75T	664	645	803	3	3	395	1670	4.23
	1	256	R2	Y	12	12	B	T	N	Y	B	-	N	XC6VLX75T	704	684	806	3	3	395	1670	4.23

Table 7: **Virtex-6 FPGA Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation[1]	Variable Point Size	Input Data Width	Phase Factor Width	Scaling Type[2]	Rounding Mode[3]	Output Ordering[4]	Cyclic Prefix Insertion	Memory Type[5]	Stages Using Block RAM	Optimize for Speed[6]	Xilinx Part[7]	LUT/FF Pairs	LUTs	FFs	18k Block RAMs[8]	XtremeDSP Slices	Max Clock Frequency[9]	Latency (cycles)[10]	Latency (µs)[11]
CT Scanners	1	1k	Str	N	24	24	S	C	R	N	-	3	Y	XC6VLX75T	3174	3127	4848	8	62	395	2179	5.52
	1	1k	Str	N	24	24	S	C	R	N	-	3	N	XC6VLX75T	3951	3878	5331	8	32	395	2167	5.49
	1	1k	Str	N	24	24	S	C	R	N	-	1	Y	XC6VLX75T	3909	3848	4885	2	62	395	2179	5.52
	1	1k	Str	N	24	24	S	C	R	N	-	5	Y	XC6VLX75T	3056	3003	4829	14	62	395	2179	5.52
	1	1k	Str	N	24	24	S	C	N	N	-	3	Y	XC6VLX75T	3284	3214	4991	11	62	395	3207	8.12
	1	1k	Str	N	24	24	U	C	N	N	-	3	Y	XC6VLX75T	3317	3262	5431	12	62	395	3203	8.11
	1	1k	Str	Y	24	24	S	C	N	N	-	3	Y	XC6VLX75T	4035	3955	5949	11	62	395	3216	8.14
	1	1k	Str	Y	16	24	U	C	R	N	-	3	Y	XC6VLX75T	3147	3070	4844	6	52	395	2181	5.52
	1	1k	Str	N	16	16	S	C	R	N	-	3	Y	XC6VLX75T	2250	2207	3307	4	40	395	2171	5.50
	1	1k	Str	N	16	16	S	C	R	N	-	3	N	XC6VLX75T	3232	3196	4173	4	12	395	2171	5.50
	1	1k	Str	N	16	16	S	C	N	N	-	3	Y	XC6VLX75T	2336	2285	3434	6	40	395	3199	8.10
	1	1k	Str	N	16	16	U	C	N	N	-	3	Y	XC6VLX75T	2448	2389	3892	8	40	395	3195	8.09
	1	1k	Str	N	34	34	S	C	N	N	-	3	Y	XC6VLX75T	4673	4577	7335	12	94	360	3223	8.95
	1	1k	Str	N	34	34	U	C	N	N	-	3	Y	XC6VLX75T	4715	4616	7919	14	102	339	3223	9.51
	1	1k	Str	N	34	34	B	C	N	N	-	3	Y	XC6VLX75T	4039	3974	6207	14	102	388	3225	8.31
	1	4k	Str	N	24	24	S	C	N	N	-	5	Y	XC6VLX75T	4109	4006	6207	30	78	395	12445	31.51
	1	4k	Str	N	24	24	U	C	N	N	-	5	Y	XC6VLX75T	4202	4151	6953	36	78	395	12441	31.50
	1	8k	Str	N	24	24	S	C	N	N	-	6	Y	XC6VLX75T	4596	4473	6906	56	90	366	24748	67.62
	1	8k	Str	N	24	24	U	C	N	N	-	6	Y	XC6VLX75T	4884	4801	7934	69	94	290	24746	85.33
U[12]	1	512	R2L	Y	24	24	S	C	N	N	B	-	N	XC6VLX75T	871	852	1224	4	4	395	5800	14.68
Test	1	8k	Str	Y	24	24	U	C	N	N	-	6	Y	XC6VLX75T	6048	5941	9392	69	94	296	24758	83.64
	1	8k	Str	Y	24	24	U	C	N	N	-	6	N	XC6VLX75T	7809	7708	10739	69	51	353	24745	70.10
	1	16k	Str	Y	24	24	U	C	N	N	-	7	Y	XC6VLX75T	6596	6514	10160	132	98	290	49341	170.14
	1	16k	Str	Y	24	24	U	C	N	N	-	7	N	XC6VLX75T	8555	8435	11569	132	51	318	49327	155.12
	1	256	R4	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	3142	3098	4285	16	40	395	1411	3.57
	1	1k	R4	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	3281	3227	4432	18	40	395	5529	14.00
	1	4k	R4	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	3361	3312	4534	37	40	395	22703	57.48
	1	256	R2	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	1952	1912	2616	8	12	395	2225	5.63
	1	1k	R2	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	2028	1987	2685	10	12	395	9427	23.87
	1	4k	R2	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	2068	2033	2741	31	12	395	41205	104.32
	1	256	R2L	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	1819	1781	2509	6	6	395	3169	8.02
	1	1k	R2L	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	1870	1843	2572	10	6	395	14441	36.56
	1	4k	R2L	N	32	24	F	-	R	N	B	-	Y	XC6VLX75T	1929	1893	2619	31	6	395	65649	166.20
	1	256	Str	N	32	24	F	-	R	N	B	1	Y	XC6VLX75T	3441	3361	5162	4	46	395	885	2.24
	1	1k	Str	N	32	24	F	-	R	N	B	3	Y	XC6VLX75T	4239	4146	6517	12	66	395	3209	8.12
	1	4k	Str	N	32	24	F	-	R	N	B	5	Y	XC6VLX75T	5034	4963	7963	36	86	366	12445	34.00

Table 7: **Virtex-6 FPGA Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation(1)	Variable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	LUT/FF Pairs	LUTs	FFs	18k Block RAMs(8)	XtremeDSP Slices	Max Clock Frequency(9)	Latency (cycles)(10)	Latency (µs)(11)
Radar	1	1k	R4	Y	16	16	S	C	N	N	B	-	N	XC6VLX75T	1978	1972	2451	7	9	395	3446	8.72
	1	1k	R4	Y	16	16	S	C	N	N	B	-	Y	XC6VLX75T	1590	1570	2145	7	20	395	3451	8.74
	1	1k	R4	Y	16	16	B	C	N	N	B	-	N	XC6VLX75T	2064	2047	2475	7	9	395	3446	8.72
	1	1k	R4	Y	20	16	S	C	N	N	B	-	N	XC6VLX75T	2355	2338	2877	11	9	395	3446	8.72
	1	1k	R4	Y	20	16	B	C	N	N	B	-	N	XC6VLX75T	2415	2398	2901	11	9	380	3446	9.07
	1	32k	R4	Y	16	16	S	C	N	N	B	-	N	XC6VLX75T	2488	2450	2838	84	9	324	131256	405.11
	1	32k	R4	Y	16	16	B	C	N	N	B	-	N	XC6VLX75T	2547	2510	2850	84	9	303	131256	433.19
	1	32k	Str	Y	16	16	U	C	N	N	-	10	N	XC6VLX75T	8153	8027	10696	201	31	235	98497	419.14

1. Implementations: Str = Pipelined, Streaming I/O; R4 = Radix-4, Burst I/O; R2 = Radix-2, Burst I/O; R2L = Radix-2 Lite, Burst I/O.
2. Scaling types: S = scaled; U = unscaled; B = block floating-point; F = single precision floating-point.
3. Rounding modes: C = convergent rounding; T = truncation.
4. Output ordering: N = Natural Order; R = Bit/Digit Reversed Order.
5. Memory types: B = block RAM, D = distributed RAM. Applies to data and phase factor storage in Burst I/O architectures, and to the output reorder buffer in the Pipelined, Streaming I/O architecture.
6. Optimize for Speed using XtremeDSP slices in both Complex Multipliers (4-multiplier structure) and Butterfly Arithmetic.
7. The -1 speedgrade was used in all cases
8. Virtex-6 FPGAs have 18K block RAMs that may be packed in pairs to form 36K block RAMs. map reports the number of 36K block RAMs + 18K block RAMs, which may not match the number of 18K block RAMs given here.
9. Area and maximum clock frequencies are provided as a guide. They may vary with the amount of other logic in the FPGA device, tools options, and other releases of Xilinx implementation tools. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.
10. Latency in clock cycles for the largest transform size.
11. Latency in microseconds for the largest transform size, when running at the maximum achievable clock frequency.
12. Ultrasound.

Virtex-5 FPGA Family

Table 8 shows performance and resource usage numbers for Virtex-5 FPGAs. A range of FFT cores is shown for several typical applications: Baseband 3GPP LTE, Baseband OFDM, CT scanners, Ultrasound, Test and measurement, and Radar. The parameters for each core are shown in Table 8. None of the optional pins (CE, SCLR, OVFLO) are used. Hybrid RAM is not used. The performance and resource usage numbers were produced using ISE 11.4 software, with speed file version "PRODUCTION 1.65 2009-04-27."

Table 8: **Virtex-5 Family Performance and Resource Utilization**

Application	Channels	Point size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for speed(6)	Xilinx Part(7)	LUT/FF Paris	LUTs	FFs	18k Block RAMs(8)	XtremeDSP Slices	Max Clock Frequency(9)	Latency (Clock Cycles)(10)	Latency (µs)(11)
Baseband 3GPP LTE	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	1179	860	1031	3	2	403	12453	30.90
	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC5VSX95T	1181	818	1033	3	3	395	12473	31.58
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	1287	976	1084	5	2	380	26804	70.54
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC5VSX95T	1288	934	1086	5	3	431	26826	62.24
	4	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	2772	1906	2603	9	8	374	12453	33.30
	8	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VLX330	4889	3302	4699	17	16	290	12453	42.94
	4	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	2906	2034	2674	17	8	360	26804	74.46
	8	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC5VLX330	5026	2776	4794	33	16	290	26804	92.43
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	1447	1114	1271	3	3	366	7364	20.12
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC5VSX95T	1253	897	1100	3	6	417	7364	17.66
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	1539	1205	1333	5	3	374	15575	41.64
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC5VSX95T	1336	954	1164	5	6	410	15564	37.96
	2	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	2318	1799	2102	5	6	347	7364	21.22
	4	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	4061	3169	3764	9	12	339	7364	21.72
	8	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VLX330	7544	5909	7088	17	24	276	7364	26.68
	2	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	2475	1943	2202	9	6	360	15575	43.26
	4	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VSX95T	4316	3419	3940	17	12	332	15575	46.91
	8	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC5VLX330	8034	6371	7416	33	24	227	15575	68.61
OFDM	1	256	R2	Y	12	12	S	T	N	Y	D	-	N	XC5VSX95T	1142	972	867	0	3	380	1652	4.35
	1	256	R2	Y	12	12	S	T	N	Y	B	-	N	XC5VSX95T	970	719	853	3	3	374	1670	4.47
	1	256	R2	Y	12	12	B	T	N	Y	B	-	N	XC5VSX95T	1001	758	856	3	3	374	1670	4.47

Table 8: **Virtex-5 Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point size	Implementation[1]	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type[2]	Rounding Mode[3]	Output Ordering[4]	Cyclic Prefix Insertion	Memory Type[5]	Stages Using Block RAM	Optimize for speed[6]	Xilinx Part[7]	LUT/FF Paris	LUTs	FFs	18k Block RAMs[8]	XtremeDSP Slices	Max Clock Frequency[9]	Latency (Clock Cycles)[10]	Latency (µs)[11]
CT Scanners	1	1k	Str	N	24	24	S	C	R	N	-	3	Y	XC5VSX95T	5034	3936	4590	8	62	431	2179	5.06
	1	1k	Str	N	24	24	S	C	R	N	-	3	N	XC5VSX95T	5495	4600	5077	8	32	380	2167	5.70
	1	1k	Str	N	24	24	S	C	R	N	-	1	Y	XC5VSX95T	5653	4742	4627	2	62	395	2179	5.52
	1	1k	Str	N	24	24	S	C	R	N	-	5	Y	XC5VSX95T	4904	3708	4571	14	62	431	2179	5.06
	1	1k	Str	N	24	24	S	C	N	N	-	3	Y	XC5VSX95T	5170	4037	4733	11	62	431	3207	7.44
	1	1k	Str	N	24	24	U	C	N	N	-	3	Y	XC5VSX95T	5595	4115	5151	12	62	403	3203	7.95
	1	1k	Str	Y	24	24	S	C	N	N	-	3	Y	XC5VSX95T	6188	4854	5494	11	62	366	3216	8.79
	1	1k	Str	Y	16	24	U	C	R	N	-	3	Y	XC5VSX95T	4997	3853	4447	6	52	417	2181	5.23
	1	1k	Str	N	16	16	S	C	R	N	-	3	Y	XC5VSX95T	3468	2712	3129	4	40	445	2171	4.88
	1	1k	Str	N	16	16	S	C	R	N	-	3	N	XC5VSX95T	4625	3791	4222	4	12	366	2171	5.93
	1	1k	Str	N	16	16	S	C	N	N	-	3	Y	XC5VSX95T	3579	2797	3256	6	40	445	3199	7.19
	1	1k	Str	N	16	16	U	C	N	N	-	3	Y	XC5VSX95T	4048	2972	3692	8	40	431	3195	7.41
	1	1k	Str	N	34	34	S	C	N	N	-	3	Y	XC5VSX95T	7574	5924	6977	12	94	380	3223	8.48
	1	1k	Str	N	34	34	U	C	N	N	-	3	Y	XC5VSX95T	8141	5971	7539	14	102	366	3223	8.81
	1	1k	Str	N	34	34	B	C	N	N	-	3	Y	XC5VSX95T	8551	6287	7742	14	102	353	3225	9.14
	1	4k	Str	N	24	24	S	C	N	N	-	5	Y	XC5VSX95T	6378	4890	5845	30	78	395	12445	31.51
	1	4k	Str	N	24	24	U	C	N	N	-	5	Y	XC5VSX95T	7156	5154	6545	36	78	360	12441	34.56
	1	8k	Str	N	24	24	S	C	N	N	-	6	Y	XC5VSX95T	7168	5440	6492	56	90	332	24748	74.54
	1	8k	Str	N	24	24	U	C	N	N	-	6	Y	XC5VSX95T	8220	5855	7460	69	94	310	24746	79.83
U[12]	1	512	R2L	Y	24	24	S	C	N	N	B	-	N	XC5VSX95T	1316	906	1224	4	4	410	5800	14.15
Test	1	8k	Str	Y	24	24	U	C	N	N	-	6	Y	XC5VSX95T	9786	7105	8646	69	94	318	24758	77.86
	1	8k	Str	Y	24	24	U	C	N	N	-	6	N	XC5VSX95T	11178	8649	9885	69	50	318	24745	77.81
	1	16k	Str	Y	24	24	U	C	N	N	-	7	Y	XC5VSX95T	10635	7771	9337	132	98	255	49341	193.49
	1	16k	Str	Y	24	24	U	C	N	N	-	7	N	XC5VSX95T	11979	9434	10618	132	50	262	49327	188.27
	1	256	R4	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	4434	2771	4243	16	40	339	1411	4.16
	1	1k	R4	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	4621	2900	4401	18	40	374	5529	14.78
	1	4k	R4	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	4673	3034	4494	37	40	353	22703	64.31
	1	256	R2	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2751	1878	2626	8	12	388	2225	5.73
	1	1k	R2	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2878	1942	2697	10	12	374	9427	25.21
	1	4k	R2	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2936	2059	2757	31	12	374	41205	110.17
	1	256	R2L	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2641	1698	2511	6	6	374	3169	8.47
	1	1k	R2L	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2724	1721	2574	10	6	339	14441	42.60
	1	4k	R2L	N	32	24	F	-	R	N	B	-	Y	XC5VSX95T	2793	1886	2621	31	6	339	65649	193.65
	1	256	Str	N	32	24	F	-	R	N	B	1	Y	XC5VSX95T	5377	4205	4983	4	46	395	885	2.24
	1	1k	Str	N	32	24	F	-	R	N	B	3	Y	XC5VSX95T	6715	5105	6209	12	66	366	3209	8.77
	1	4k	Str	N	32	24	F	-	R	N	B	5	Y	XC5VSX95T	8242	6117	7512	36	86	360	12445	34.57

Table 8: **Virtex-5 Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for speed(6)	Xilinx Part(7)	LUT/FF Paris	LUTs	FFs	18k Block RAMs(8)	XtremeDSP Slices	Max Clock Frequency(9)	Latency (Clock Cycles)(10)	Latency (µs)(11)
Radar	1	1k	R4	Y	16	16	S	C	N	N	B	-	N	XC5VSX95T	2813	2220	2614	7	9	347	3446	9.93
	1	1k	R4	Y	16	16	S	C	N	N	B	-	Y	XC5VSX95T	2259	1451	2134	7	20	388	3451	8.89
	1	1k	R4	Y	16	16	B	C	N	N	B	-	N	XC5VSX95T	2887	2325	2614	7	9	347	3446	9.93
	1	1k	R4	Y	20	16	S	C	N	N	B	-	N	XC5VSX95T	3158	2532	3022	11	9	347	3446	9.93
	1	1k	R4	Y	20	16	B	C	N	N	B	-	N	XC5VSX95T	3297	2637	3022	11	9	324	3446	10.64
	1	32k	R4	Y	16	16	S	C	N	N	B	-	N	XC5VSX95T	3332	2753	2966	84	9	283	131256	463.80
	1	32k	R4	Y	16	16	B	C	N	N	B	-	N	XC5VSX95T	3369	2832	2978	84	9	310	131256	423.41
	1	32k	Str	Y	16	16	U	C	N	N	-	10	N	XC5VSX95T	11687	9175	9727	201	31	220	98497	447.71

1. Implementations: Str = Pipelined, Streaming I/O; R4 = Radix-4, Burst I/O; R2 = Radix-2, Burst I/O; R2L = Radix-2 Lite, Burst I/O.
2. Scaling types: S = scaled; U = unscaled; B = block floating-point; F = single precision floating-point.
3. Rounding modes: C = convergent rounding; T = truncation.
4. Output ordering: N = Natural Order; R = Bit/Digit Reversed Order.
5. Memory types: B = block RAM, D = distributed RAM. Applies to data and phase factor storage in Burst I/O architectures and to the output reorder buffer in the Pipelined, Streaming I/O architecture.
6. Optimize for Speed using XtremeDSP slices in both Complex Multipliers (4-multiplier structure) and Butterfly Arithmetic.
7. The -1 speedgrade was used in all cases
8. Virtex-5 FPGAs have 18K block RAMs that may be packed in pairs to form 36K block RAMs. map reports the number of 36K block RAMs + 18K block RAMs, which may not match the number of 18K block RAMs given here.
9. Area and maximum clock frequencies are provided as a guide. They may vary with the amount of other logic in the FPGA device, tools options, and other releases of Xilinx implementation tools. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.
10. Latency in clock cycles for the largest transform size.
11. Latency in microseconds for the largest transform size, when running at the maximum achievable clock frequency.
12. Ultrasound.

Spartan-6 Family

Table 9 shows performance and resource usage numbers for Spartan-6 FPGAs. A range of FFT cores is shown for several typical applications: Baseband 3GPP LTE, Baseband OFDM, CT scanners, Ultrasound, Test and measurement, and Radar. The parameters for each core are shown in Table 9. Some rows of the table are grayed-out to indicate that these cores would not fit on the device due to FPGA resource requirements (typically insufficient I/O pins to route all core signals outside the device). None of the optional pins (CE, SCLR, OVFLO) are used. Hybrid RAM is not used. The performance and resource usage numbers were produced using ISE 12.1 software, with speed file version "PRELIMINARY 1.08b 2010-03-16."

Table 9: **Spartan-6 Family Performance and Resource Utilization**

Application	Channels	Point Size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	LUT/FF Paris	LUTs	FFs	9k Block RAMs(8)	XtremeDSP slices	Max clock frequency(9)	Latency (clock cycles)(10)	Latency(µs)(11)
Baseband 3GPP LTE	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	838	818	1032	6	2	251	12453	49.61
	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC6SLX150T	819	800	1034	6	3	188	12473	66.35
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	965	936	1087	9	2	244	26804	109.85
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC6SLX150T	954	922	1089	9	3	236	26826	113.67
	4	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	2281	1378	2610	19	8	195	12453	63.86
	8	1k	R2L	Y	16	16	S	C	N	Y	B	-	N									
	4	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	2380	1480	2675	33	8	210	26804	127.64
	8	2k	R2L	Y	16	16	S	C	N	Y	B	-	N									
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	1038	1014	1204	6	3	236	7364	31.20
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC6SLX150T	916	884	1090	6	6	172	7364	42.81
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	1125	1095	1266	9	3	236	15575	66.00
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC6SLX150T	982	946	1154	9	6	251	15564	62.01
	2	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	1621	1558	1978	10	6	228	7364	32.30
	4	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	3132	2357	3526	18	12	210	7364	35.07
	8	1k	R2	Y	16	16	S	C	N	Y	B	-	N									
	2	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	1748	1708	2078	17	6	219	15575	71.12
	4	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC6SLX150T	3344	2567	3701	33	12	210	15575	74.17
	8	2k	R2	Y	16	16	S	C	N	Y	B	-	N									
OFDM	1	256	R2	Y	12	12	S	T	N	Y	D	-	N	XC6SLX150T	878	858	817	0	3	251	1652	6.58
	1	256	R2	Y	12	12	S	T	N	Y	B	-	N	XC6SLX150T	681	657	803	3	3	219	1670	7.63
	1	256	R2	Y	12	12	B	T	N	Y	B	-	N	XC6SLX150T	721	696	806	3	3	219	1670	7.63

Table 9: **Spartan-6 Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	LUT/FF Paris	LUTs	FFs	9k Block RAMs(8)	XtremeDSP slices	Max clock frequency(9)	Latency (clock cycles)(10)	Latency(µs)(11)
CT scanners	1	1k	Str	N	24	24	S	C	R	N	-	3	Y	XC6SLX150T	3574	3431	5427	9	94	156	2203	14.12
	1	1k	Str	N	24	24	S	C	R	N	-	3	N	XC6SLX150T	5481	5401	7151	9	48	196	2215	11.30
	1	1k	Str	N	24	24	S	C	R	N	-	1	Y	XC6SLX150T	4352	4233	5466	3	94	149	2203	14.79
	1	1k	Str	N	24	24	S	C	R	N	-	5	Y	XC6SLX150T	3441	3305	5409	15	94	156	2203	14.12
	1	1k	Str	N	24	24	S	C	N	N	-	3	Y	XC6SLX150T	3619	3505	5570	15	94	165	3231	19.58
	1	1k	Str	N	24	24	U	C	N	N	-	3	Y	XC6SLX150T	3749	3618	6108	17	94	165	3227	19.56
	1	1k	Str	Y	24	24	S	C	N	N	-	3	Y	XC6SLX150T	4468	4323	6528	15	94	172	3240	18.84
	1	1k	Str	Y	16	24	U	C	R	N	-	3	Y	XC6SLX150T	3511	3395	5305	7	80	180	2202	12.23
	1	1k	Str	N	16	16	S	C	R	N	-	3	Y	XC6SLX150T	2308	2206	3298	6	40	188	2171	11.55
	1	1k	Str	N	16	16	S	C	R	N	-	3	N	XC6SLX150T	2796	2712	3633	6	16	219	2159	9.86
	1	1k	Str	N	16	16	S	C	N	N	-	3	Y	XC6SLX150T	2415	2298	3425	10	40	196	3199	16.32
	1	1k	Str	N	16	16	U	C	N	N	-	3	Y	XC6SLX150T	2606	2483	4054	13	52	165	3204	19.42
	1	1k	Str	N	34	34	S	C	N	N	-	3	Y	XC6SLX150T	5083	4919	7908	18	126	141	3251	23.06
	1	1k	Str	N	34	34	U	C	N	N	-	3	Y	XC6SLX150T	5052	4898	8386	21	126	134	3247	24.23
	1	1k	Str	N	34	34	B	C	N	N	-	3	Y	XC6SLX150T	5455	5224	8585	21	126	110	3253	29.57
	1	4k	Str	N	24	24	S	C	N	N	-	5	Y	XC6SLX150T	4472	4338	6930	52	118	110	12475	113.41
	1	4k	Str	N	24	24	U	C	N	N	-	5	Y	XC6SLX150T	4749	4614	7830	64	118	149	12471	83.70
	1	8k	Str	N	24	24	S	C	N	N	-	6	Y	XC6SLX150T	5105	4893	7780	101	138	149	24784	166.34
	1	8k	Str	N	24	24	U	C	N	N	-	6	Y	XC6SLX150T	5503	5355	9030	128	146	141	24785	175.78
U(12)	1	512	R2L	Y	24	24	S	C	N	N	B	-	N	XC6SLX150T	930	902	1299	6	8	188	5854	31.14

Table 9: **Spartan-6 Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation[1]	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type[2]	Rounding Mode[3]	Output Ordering[4]	Cyclic Prefix Insertion	Memory Type[5]	Stages Using Block RAM	Optimize for Speed[6]	Xilinx Part[7]	LUT/FF Paris	LUTs	FFs	9k Block RAMs[8]	XtremeDSP slices	Max clock frequency[9]	Latency (clock cycles)[10]	Latency(µs)[11]
Test	1	8k	Str	Y	24	24	U	C	N	N	-	6	Y	XC6SLX150T	6811	6633	10488	128	146	125	24797	198.38
	1	8k	Str	Y	24	24	U	C	N	N	-	6	N	XC6SLX150T	10086	9921	13303	128	80	156	24819	159.10
	1	16k	Str	Y	24	24	U	C	N	N	-	7	Y	XC6SLX150T	7341	7211	11248	254	150	134	49380	368.51
	1	16k	Str	Y	24	24	U	C	N	N	-	7	N	XC6SLX150T	10799	10644	14131	254	80	125	49401	395.21
	1	256	R4	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	3374	3301	4714	16	64	149	1435	9.63
	1	1k	R4	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	3499	3419	4870	22	64	141	5559	39.43
	1	4k	R4	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	3608	3529	4972	66	64	125	22739	181.91
	1	256	R2	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	2116	2043	2778	8	20	180	2273	12.63
	1	1k	R2	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	2182	2111	2840	18	20	156	9487	60.81
	1	4k	R2	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	2182	2127	2892	58	20	156	41277	264.60
	1	256	R2L	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	1875	1832	2608	6	10	219	3175	14.50
	1	1k	R2L	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	1973	1904	2669	18	10	244	14447	59.21
	1	4k	R2L	N	32	24	F	-	R	N	B	-	Y	XC6SLX150T	2017	1943	2716	58	10	212	65655	309.69
	1	256	Str	N	32	24	F	-	R	N	B	1	Y	XC6SLX150T	3762	3639	5679	4	70	156	903	5.79
	1	1k	Str	N	32	24	F	-	R	N	B	3	Y	XC6SLX150T	4671	4522	7236	18	102	149	3236	21.72
	1	4k	Str	N	32	24	F	-	R	N	B	5	Y	XC6SLX150T	5600	5457	8893	62	134	125	12481	99.85
Radar	1	1k	R4	Y	16	16	S	C	N	N	B	-	N	XC6SLX150T	1995	1961	2451	10	9	203	3446	16.98
	1	1k	R4	Y	16	16	S	C	N	N	B	-	Y	XC6SLX150T	1610	1564	2143	10	20	196	3451	17.61
	1	1k	R4	Y	16	16	B	C	N	N	B	-	N	XC6SLX150T	2067	2023	2475	10	9	125	3446	27.57
	1	1k	R4	Y	20	16	S	C	N	N	B	-	N	XC6SLX150T	2512	2472	3003	14	18	196	3466	17.68
	1	1k	R4	Y	20	16	B	C	N	N	B	-	N	XC6SLX150T	2551	2525	3027	14	18	203	3466	17.07
	1	32k	R4	Y	16	16	S	C	N	N	B	-	N	XC6SLX150T	2461	2414	2838	164	9	149	131256	880.91
	1	32k	R4	Y	16	16	B	C	N	N	B	-	N	XC6SLX150T	2512	2472	2850	164	9	141	131256	930.89
	1	32k	Str	Y	16	16	U	C	N	N	-	10	N	XC6SLX150T	8431	8268	10543	389	40	102	98515	965.83

1. Implementations: Str = Pipelined, Streaming I/O; R4 = Radix-4, Burst I/O; R2 = Radix-2, Burst I/O; R2L = Radix-2 Lite, Burst I/O.
2. Scaling types: S = scaled; U = unscaled; B = block floating-point; F = single precision floating-point.
3. Rounding modes: C = convergent rounding; T = truncation.
4. Output ordering: N = Natural Order; R = Bit/Digit Reversed Order.
5. Memory types: B = block RAM, D = distributed RAM. Applies to data and phase factor storage in Burst I/O architectures, and to the output reorder buffer in the Pipelined, Streaming I/O architecture.
6. Optimize for Speed using XtremeDSP slices in both Complex Multipliers (4-multiplier structure) and Butterfly Arithmetic.
7. The -2 speedgrade was used in all cases
8. Spartan-6 FPGAs have 9K block RAMs that may be packed in pairs to form 18K block RAMs. map reports the number of 18K block RAMs + 9K block RAMs, which may not match the number of 9K block RAMs given here.
9. Area and maximum clock frequencies are provided as a guide. They may vary with the amount of other logic in the FPGA device, tools options, and other releases of Xilinx implementation tools. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.
10. Latency in clock cycles for the largest transform size.
11. Latency in microseconds for the largest transform size, when running at the maximum achievable clock frequency.
12. Ultrasound.

Spartan-3A DSP Family

Table 10 shows performance and resource usage numbers for Spartan-3A DSP FPGAs. A range of FFT cores is shown for several typical applications: Baseband 3GPP LTE, Baseband OFDM, CT scanners, Ultrasound, Test and measurement, and Radar. The parameters for each core are shown in Table 9. Some rows of the table are grayed-out to indicate that these cores would not fit on the device due to FPGA resource requirements (typically insufficient I/O pins to route all core signals outside the device). None of the optional pins (CE, SCLR, OVFLO) are used. Hybrid RAM is not used. The performance and resource usage numbers were produced using ISE 11.4 software, with speed file version "PRODUCTION 1.33 2009-04-27."

Table 10: **Spartan-3A DSP Family Performance and Resource Utilization**

Application	Channels	Point Size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	Slices	LUTs	FFs	18k Block RAMs	XtremeDSP slices	Max clock frequency(8)	Latency (clock cycles)(9)	Latency(µs)(10)
Baseband 3GPP LTE	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	736	1004	1062	3	2	203	12453	61.34
	1	1k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC3SD3400A	741	972	1064	3	3	203	12473	61.44
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	811	1160	1117	5	2	172	26804	155.84
	1	2k	R2L	Y	16	16	S	C	N	Y	B	-	Y	XC3SD3400A	817	1130	1119	5	3	180	26826	149.03
	4	1k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	1710	2233	2640	9	8	188	12453	66.24
	8	1k	R2L	Y	16	16	S	C	N	Y	B	-	N									
	4	2k	R2L	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	1805	2395	2718	17	8	196	26804	136.76
	8	2k	R2L	Y	16	16	S	C	N	Y	B	-	N									
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	909	1397	1273	3	3	203	7364	36.28
	1	1k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC3SD3400A	838	1198	1160	3	6	203	7354	36.23
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	969	1515	1339	5	3	203	15575	76.72
	1	2k	R2	Y	16	16	S	C	N	Y	B	-	Y	XC3SD3400A	885	1280	1227	5	6	212	15564	73.42
	2	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	1409	2245	2049	5	6	188	7364	39.17
	4	1k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	2468	3874	3597	9	12	180	7364	40.91
	8	1k	R2	Y	16	16	S	C	N	Y	B	-	N									
	2	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	1501	2412	2155	9	6	188	15575	82.85
	4	2k	R2	Y	16	16	S	C	N	Y	B	-	N	XC3SD3400A	2618	4144	3779	17	12	180	15575	86.53
	8	2k	R2	Y	16	16	S	C	N	Y	B	-	N									
OFDM	1	256	R2	Y	12	12	S	T	N	Y	D	-	N	XC3SD3400A	1197	2063	1433	0	3	212	1679	7.92
	1	256	R2	Y	12	12	S	T	N	Y	B	-	N	XC3SD3400A	604	945	838	3	3	212	1670	7.88
	1	256	R2	Y	12	12	B	T	N	Y	B	-	N	XC3SD3400A	643	945	838	3	3	196	1670	8.52

Table 10: **Spartan-3A DSP Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation[1]	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type[2]	Rounding Mode[3]	Output Ordering[4]	Cyclic Prefix Insertion	Memory Type[5]	Stages Using Block RAM	Optimize for Speed[6]	Xilinx Part[7]	Slices	LUTs	FFs	18k Block RAMs	XtremeDSP slices	Max clock frequency[8]	Latency (clock cycles)[9]	Latency(µs)[10]
CT scanners	1	1k	Str	N	24	24	S	C	R	N	-	3	Y	XC3SD3400A	3408	5076	5167	7	94	203	2203	10.85
	1	1k	Str	N	24	24	S	C	R	N	-	3	N	XC3SD3400A	4440	7339	6843	7	48	165	2215	13.42
	1	1k	Str	N	24	24	S	C	R	N	-	1	Y	XC3SD3400A	4373	7157	5358	2	94	134	2203	16.44
	1	1k	Str	N	24	24	S	C	R	N	-	5	Y	XC3SD3400A	3204	4521	5149	12	94	212	2203	10.39
	1	1k	Str	N	24	24	S	C	N	N	-	3	Y	XC3SD3400A	3494	5175	5310	10	94	203	3231	15.92
	1	1k	Str	N	24	24	U	C	N	N	-	3	Y	XC3SD3400A	3600	4997	5826	11	94	188	3227	17.16
	1	1k	Str	Y	24	24	S	C	N	N	-	3	Y	XC3SD3400A	4144	6147	6072	10	94	196	3240	16.53
	1	1k	Str	Y	16	24	U	C	R	N	-	3	Y	XC3SD3400A	3277	4654	4908	5	80	196	2202	11.23
	1	1k	Str	N	16	16	S	C	R	N	-	3	Y	XC3SD3400A	2151	3263	3118	4	40	212	2171	10.24
	1	1k	Str	N	16	16	S	C	R	N	-	3	N	XC3SD3400A	2307	3693	3466	4	16	196	2159	11.02
	1	1k	Str	N	16	16	S	C	N	N	-	3	Y	XC3SD3400A	2230	3346	3245	6	40	219	3199	14.61
	1	1k	Str	N	16	16	U	C	N	N	-	3	Y	XC3SD3400A	2445	3443	3853	8	52	212	3204	15.11
	1	1k	Str	N	34	34	S	C	N	N	-	3	Y	XC3SD3400A	4844	7321	7546	11	126	172	3251	18.90
	1	1k	Str	N	34	34	U	C	N	N	-	3	Y	XC3SD3400A	4884	6891	8002	13	126	165	3247	19.68
	1	1k	Str	N	34	34	B	C	N	N	-	3	Y	XC3SD3400A	5280	7606	8412	13	126	165	3253	19.72
	1	4k	Str	N	24	24	S	C	N	N	-	5	Y	XC3SD3400A	4303	6261	6565	29	118	203	12475	61.45
	1	4k	Str	N	24	24	U	C	N	N	-	5	Y	XC3SD3400A	4561	6239	7419	35	118	180	12471	69.28
	1	8k	Str	N	24	24	S	C	N	N	-	6	Y									
	1	8k	Str	N	24	24	U	C	N	N	-	6	Y									
U[11]	1	512	R2L	Y	24	24	S	C	N	N	B	-	N	XC3SD3400A	831	1158	1323	3	8	188	5854	31.14

Table 10: **Spartan-3A DSP Family Performance and Resource Utilization *(Cont'd)***

Application	Channels	Point Size	Implementation(1)	Configurable Point Size	Input Data Width	Phase Factor Width	Scaling Type(2)	Rounding Mode(3)	Output Ordering(4)	Cyclic Prefix Insertion	Memory Type(5)	Stages Using Block RAM	Optimize for Speed(6)	Xilinx Part(7)	Slices	LUTs	FFs	18k Block RAMs	XtremeDSP slices	Max clock frequency(8)	Latency (clock cycles)(9)	Latency(µs)(10)
Test	1	8k	Str	Y	24	24	U	C	N	N	-	6	Y									
	1	8k	Str	Y	24	24	U	C	N	N	-	6	N	XC3SD3400A	8047	12691	12600	67	80	156	24819	159.10
	1	16k	Str	Y	24	24	U	C	N	N	-	7	Y									
	1	16k	Str	Y	24	24	U	C	N	N	-	7	N									
	1	256	R4	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	3170	4157	4679	13	64	180	1435	7.97
	1	1k	R4	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	3298	4322	4845	15	64	165	5559	33.69
	1	4k	R4	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	3367	4489	4938	37	64	180	22739	126.33
	1	256	R2	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	1893	2526	2786	7	20	196	2273	11.60
	1	1k	R2	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	1968	2654	2883	9	20	188	9487	50.46
	1	4k	R2	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	2021	2795	2957	31	20	188	41277	219.56
	1	256	R2L	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	1682	2128	2606	5	10	188	3175	16.89
	1	1k	R2L	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	1746	2200	2699	9	10	188	14447	78.85
	1	4k	R2L	N	32	24	F	-	R	N	B	-	Y	XC3SD3400A	1785	2381	2752	31	10	180	65655	364.75
	1	256	Str	N	32	24	F	-	R	N	B	1	Y	XC3SD3400A	3566	5200	5504	4	70	172	903	5.25
	1	1k	Str	N	32	24	F	-	R	N	B	3	Y	XC3SD3400A	4439	6276	6932	11	102	172	3236	18.81
	1	4k	Str	N	32	24	F	-	R	N	B	5	Y									
Radar	1	1k	R4	Y	16	16	S	C	N	N	B	-	N	XC3SD3400A	1691	2737	2441	7	9	188	3466	18.44
	1	1k	R4	Y	16	16	S	C	N	N	B	-	Y	XC3SD3400A	1529	2069	2135	7	20	188	3451	18.36
	1	1k	R4	Y	16	16	B	C	N	N	B	-	N	XC3SD3400A	1764	2873	2465	7	9	172	3446	20.03
	1	1k	R4	Y	20	16	S	C	N	N	B	-	N	XC3SD3400A	2098	3402	3017	11	18	165	3466	21.01
	1	1k	R4	Y	20	16	B	C	N	N	B	-	N	XC3SD3400A	2170	3538	3040	11	18	165	3466	21.01
	1	32k	R4	Y	16	16	S	C	N	N	B	-	N	XC3SD3400A	2085	3476	2816	84	9	156	131256	841.38
	1	32k	R4	Y	16	16	B	C	N	N	B	-	N	XC3SD3400A	2156	3617	2830	84	9	156	131256	841.38
	1	32k	Str	Y	16	16	U	C	N	N	-	10	N									

1. Implementations: Str = Pipelined, Streaming I/O; R4 = Radix-4, Burst I/O; R2 = Radix-2, Burst I/O; R2L = Radix-2 Lite, Burst I/O.
2. Scaling types: S = scaled; U = unscaled; B = block floating-point; F = single precision floating-point.
3. Rounding modes: C = convergent rounding; T = truncation.
4. Output ordering: N = Natural Order; R = Bit/Digit Reversed Order.
5. Memory types: B = block RAM, D = distributed RAM. Applies to data and phase factor storage in Burst I/O architectures, and to the output reorder buffer in the Pipelined, Streaming I/O architecture.
6. Optimize for Speed using XtremeDSP slices in both Complex Multipliers (4-multiplier structure) and Butterfly Arithmetic.
7. The -4 speedgrade was used in all cases
8. Area and maximum clock frequencies are provided as a guide. They may vary with the amount of other logic in the FPGA device, tools options, and other releases of Xilinx implementation tools. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.
9. Latency in clock cycles for the largest transform size.
10. Latency in microseconds for the largest transform size, when running at the maximum achievable clock frequency.
11. Ultrasound.

Dynamic Range Characteristics

The dynamic range characteristics are shown by performing *slot noise* tests. First, a frame of complex Gaussian noise data samples is created. An FFT is taken to acquire the spectrum of the data. To create the slot, a range of frequencies in the spectra is set to zero. To create the input slot noise data frame, the inverse FFT is taken, then the data is quantized to use the full input dynamic range. Because of the quantization, if a perfect FFT is done on the frame, the noise floor on the bottom of the slot is non-zero. The Input Data figures, which basically represent the dynamic range of the input format, display this.

This slot noise input data frame is fed to the FFT core to see how shallow the slot becomes due to the finite precision arithmetic. The depth of the slot shows the dynamic range of the FFT.

Figure 17 through Figure 26 show the effect of input data width on the dynamic range. All FFTs have the same bit width for both data and phase factors. Block floating-point arithmetic is used with rounding after the butterfly. The figures show the input data slot and the output data slot for bit widths of 24, 20, 16, 12, and 8.

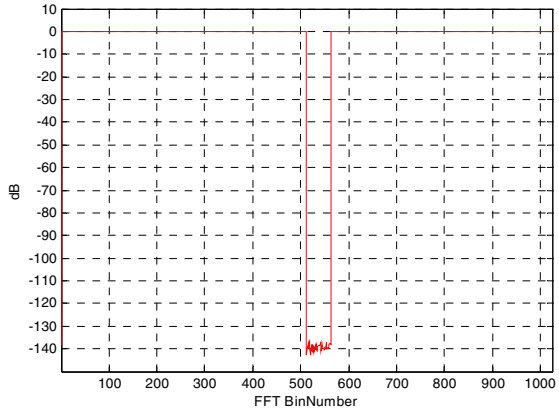


Figure 17: **Input Data: 24 Bits**

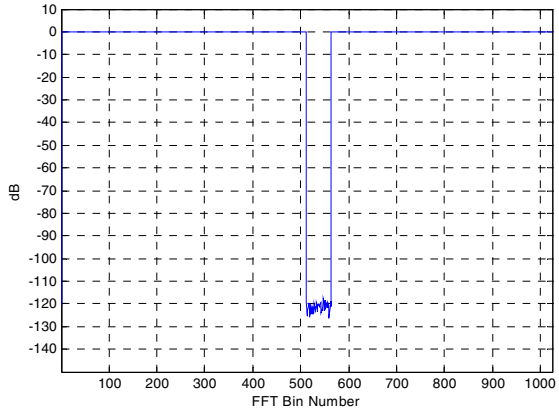


Figure 18: **FFT Core Results: 24 Bits**

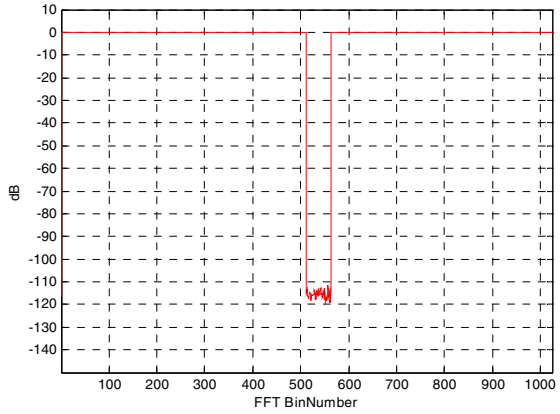


Figure 19: **Input Data: 20 Bits**

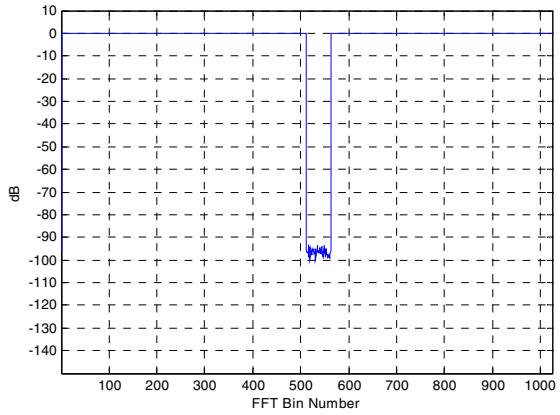


Figure 20: **FFT Core Results: 20 Bits**

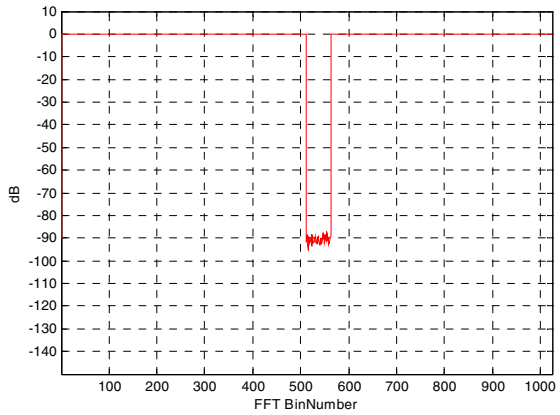


Figure 21: **Input Data: 16 Bits**

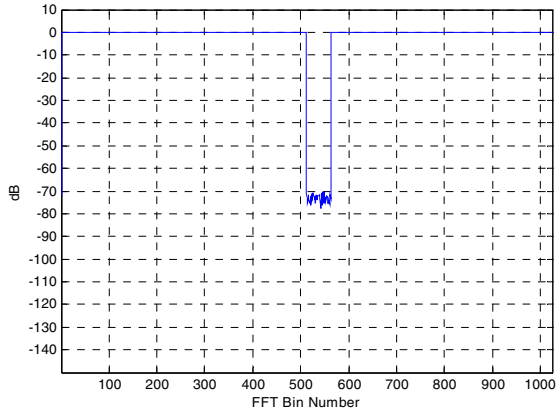


Figure 22: **FFT Core Results: 16 Bits**

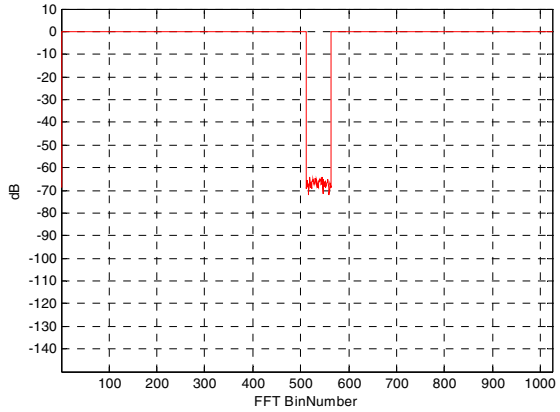


Figure 23: **Input Data: 12 Bits**

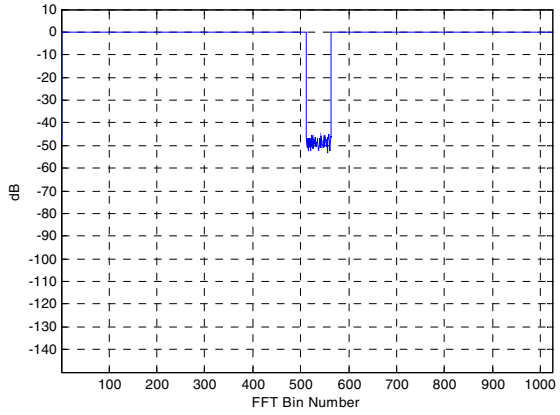


Figure 24: **FFT Core Results: 12 Bits**

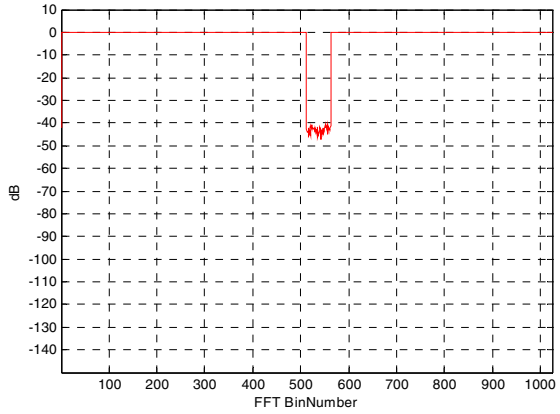


Figure 25: **Input Data: 8 Bits**

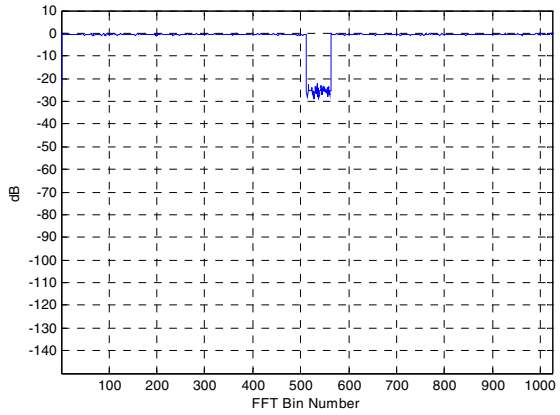


Figure 26: **FFT Core Results: 8 Bits**

There are several options available that also affect the dynamic range. Consider the arithmetic type used.

Figure 27, Figure 28, and Figure 29 display the results of using unscaled, scaled (scaling of 1/1024), and block floating-point. All three FFTs are 1024 point, Radix-4, Burst I/O transforms with 16-bit input, 16-bit phase factors, and convergent rounding.

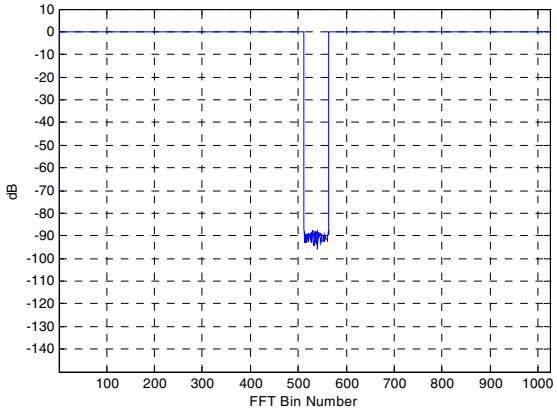


Figure 27: **Full-Precision Unscaled Arithmetic**

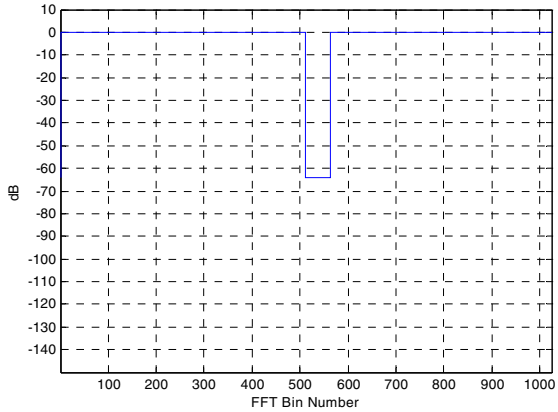


Figure 28: **Scaled (scaling of 1/N) Arithmetic**

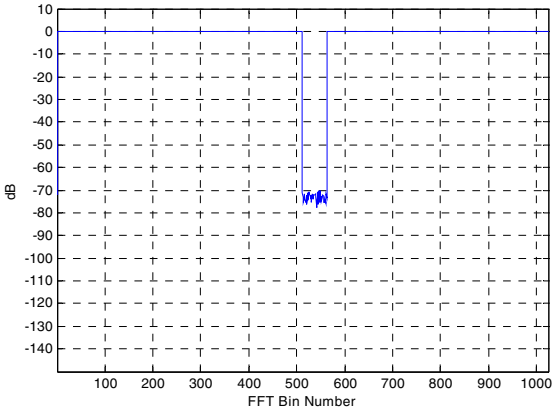


Figure 29: **Block Floating-Point Arithmetic**

After the butterfly computation, the LSBs of the data path can be truncated or rounded. The effects of these options are shown in Figure 30 and Figure 31. Both transforms are 1024 points with 16-bit data and phase factors using block floating-point arithmetic.

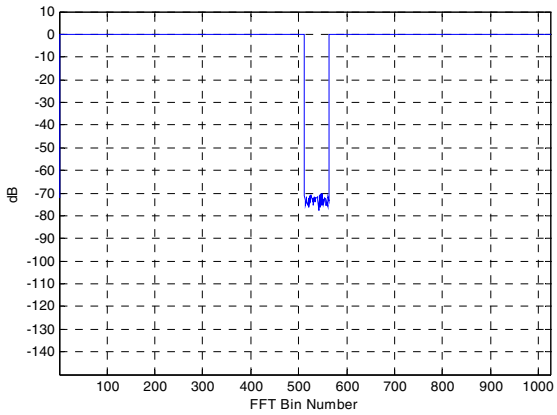


Figure 30: **Convergent Rounding**

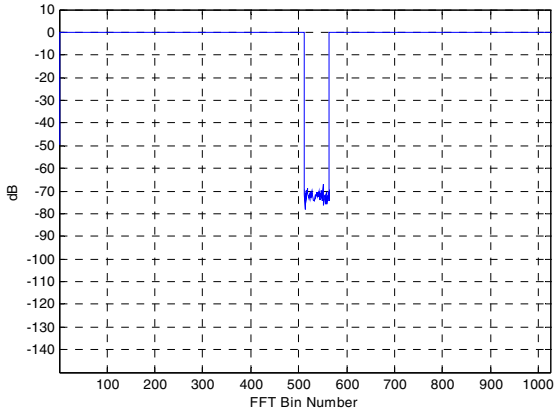


Figure 31: **Truncation**

For illustration purposes, the effect of point size on dynamic range is displayed Figure 32 through Figure 34. The FFTs in these figures use 16-bit input and phase factors along with convergent rounding and block floating-point arithmetic.

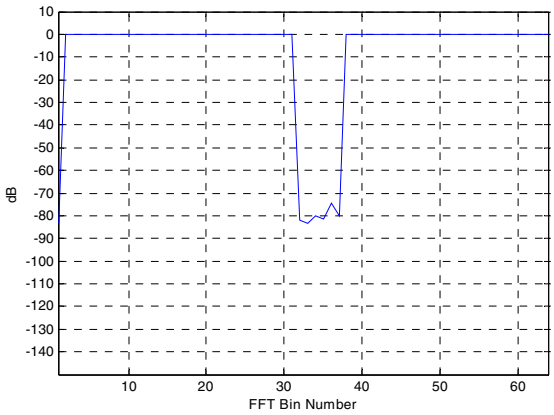


Figure 32: **64-point Transform**

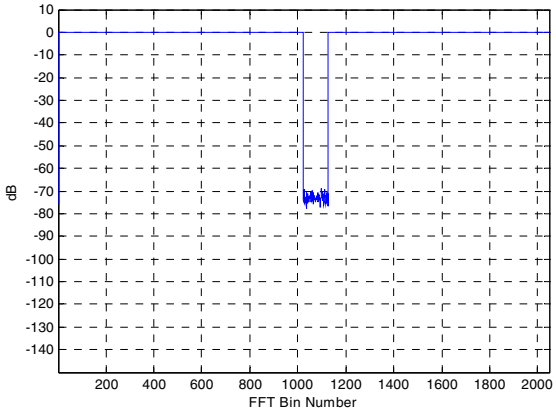


Figure 33: **2048-point Transform**

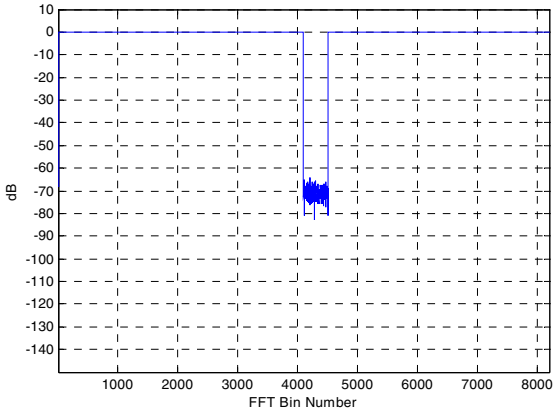


Figure 34: **8192-point Transform**

All of the preceding dynamic range plots show the results for the Radix-4, Burst I/O architecture. Figure 35 and Figure 36 show two plots for the Radix-2, Burst I/O architecture. Both use 16-bit input and phase factors along with convergent rounding and block floating-point.

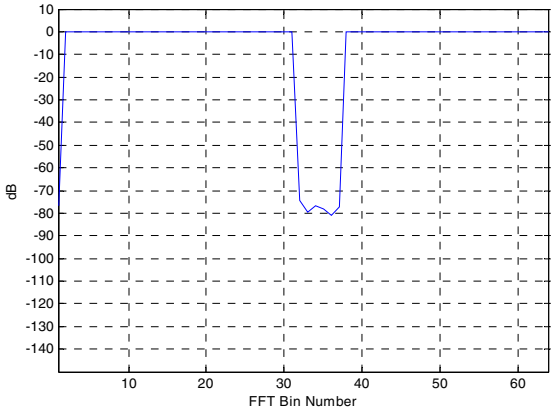


Figure 35: **64-point Radix-2 Transform**

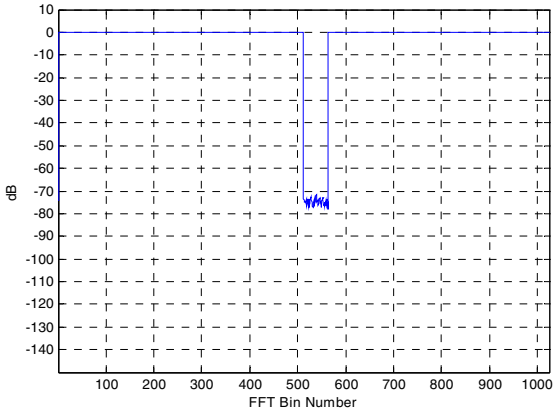


Figure 36: **1024-point Radix-2 Transform**

Simulator Support

The core has been tested with the following simulators:

- Mentor Graphics ModelSim v6.5c
- Cadence Incisive Enterprise Simulator (IES) v9.2
- ISE Simulator 12.1

References

1. 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, December 1979.
2. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.
3. Quang Hung Nguyen and Istvan Kollar, *Limited Dynamic Range of Spectrum Analysis Due To Round off Errors Of The FFT*, available at: home.mit.bme.hu/~kollar/papers/IMTC-FFT.pdf
4. I. Szolik, K. Kovac, V. Smiesko, *Influence of Digital Signal Processing on Precision of Power Quality Parameters Measurement*, available at: www.measurement.sk/2003/S1/Szolik.pdf.
5. Xilinx, Inc., *XtremeDSP DSP48A for Spartan-3A DSP FPGAs User Guide*, UG431.
6. J. W. Cooley and J. W. Tukey, *An Algorithm for the Machine Computation of Complex Fourier Series*, *Mathematics of Computation*, Vol. 19, pp. 297-301, April 1965.
7. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms and Applications Second Edition*, Maxwell Macmillan International, New York, 1992.

Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Refer to the *IP Release Notes Guide* (XTP025) for further information on this core. There will be a link to all the DSP IP and then to the relevant core being designed with.

For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

The FFT core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator software v12.1 and higher. The Xilinx CORE Generator software is bundled with all ISE Design Suite packages at no additional charge. Information about additional Xilinx® LogiCORE IP modules is available on the Xilinx IP Center.

To order Xilinx software, contact your local Xilinx sales representative.

Revision History

Date	Version	Revision
03/28/03	1.0	Xilinx release in new template.
07/14/03	2.0	Modified Figures 8 through 14, inclusive.
12/11/03	2.1	Updated to v2.1 release.
05/21/04	3.0	Updated to v3.0 release.
11/11/04	3.1	Updated document to support core v3.1 release - updated performance and resource utilization tables for Virtex-II and Virtex-II Pro FPGAs. Also added performance and resource utilization tables for Virtex-4 FPGAs.
8/31/05	3.2	Updated documentation for v3.2 core release; updated performance and resource utilization tables; updated for ISE v7.1i SP4 software.
1/11/06	3.3	Corrected table 9, XtremeDSP Slices, row 3.
11/30/06	3.4	Updated to v4.0 release.
02/15/07	4.1	Updated to v4.1 release.
10/10/07	5.0	Updated to v5.0 release.
09/19/08	6.0	Updated to v6.0 release.
06/24/09	7.0	Updated to v7.0 release.
04/19/10	7.1	Updated to v7.1 release.

Notice of Disclaimer