High-Performance 64-Point Complex FFT/IFFT V1.0.5

July 5 2000 Product Specification

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

1 Features

- Drop-in module for Virtex™ and Virtex™-E and Virtex™-EM FPGAs
- High-performance 64-point complex FFT and inverse FFT (IFFT)
- 16-bit complex input and output data
- 2's complement arithmetic
- Flexible I/O and memory interface to Virtex on-chip Block RAM
- Support for overlapping data input, transform calculation and data output operations
- Naturally ordered input and output data
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Incorporates Xilinx Smart-IP technology for maximum performance
- To be used with version 2.1i or later of the Xilinx CORE Generator System

2 General Description

The vFFT64 fast Fourier transform (FFT) Core computes a 64-point complex forward FFT or inverse FFT (IFFT). The input data is a vector of 64 complex values represented as 16-bit 2's complement numbers – 16-bits for each of the real and imaginary component of a data sample. The 64 element output vector is also represented using 16 bits for each of the real and imaginary components of an output sample. Three memory and data I/O interfaces are supported. The user interface can be configured to allow the vfft64 core to simultaneously input new data, transform data stored in memory, and to output previous results.

3 Theory of Operation

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

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

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

The Xilinx 64-point transform engine employs a Cooley-Tukey radix-4 decimation-in-frequency (DIF) FFT [1] to compute the DFT of a complex sequence. In general, this algorithm requires the calculation of columns or *ranks* of radix-4 butterflies. These radix-4 butterflies are sometimes referred to as *dragonflies.* Each processing rank consists of $N/4$ dragonflies. For $N = 64$ there are 3 dragonfly ranks, with each rank comprising 16 dragonflies.

The FFT processor input-data for the core is a vector of 64 complex samples. The real and imaginary components of each sample are represented as 16-bit 2's complement numbers. The data is stored in on-chip dual port Block RAM. The phase factors used in the FFT calculation are generated within the core. Like the input-data, the phase factors are kept to a precision of 16 bits. The complex output samples are also defined with 16 bits of precision for each of the real and imaginary components.

All of the control signals required to interface the FFT module to Block RAM are generated by the core. I/O interface signals are generated by the core to provide a flexible user interface for supplying input vectors and reading FFT result samples.

4 Finite Word Length Considerations

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

The Xilinx 64-point FFT Core scales dragonfly results by a factor of 4 on each processing pass. The *SCALE_MODE* pin can be used to force an additional scaling by one bit on the first processing pass only. The scaling results in the final output sequence being modified by the factor 1/64 when *SCALE_MODE=0* and 1/128 when *SCALE_MODE*=1.
The scaling results in the final output sequence being modified by the factor $1/sN$ where *N*=64 for the vfft64 core. Formally, the output sequence $X^{'}(k), k = 0,\ldots,N-1$ computed by the core is defined in Eq. (2)

$$X^{'}(k) = \frac{1}{sN} X(k) = \frac{1}{sN} \sum_{n=0}^{N-1} x(n) e^{-jnk2p/N} \quad k = 0,\ldots,N-1 \qquad (2)$$

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

The vfft64 core also computes the IFFT according to the following defining equation

$$x(n) = \frac{1}{sN}\sum_{k=0}^{N-1} X(k)e^{j2pnk/N} \qquad n = 0,1,\ldots,N-1 \tag{3}$$

The in-built scaling in the core accounts for the 1/N scale factor in front of the summation in Eq. (3). When *SCALE_MODE*=1, an additional scaling by a factor of ½ will be scheduled in the core. The additional scaling by 1 bit is inserted during the memory write operation of the first of the 3 processing phases.

5 Pinout

The vfft64 symbol is shown in Figure 1.

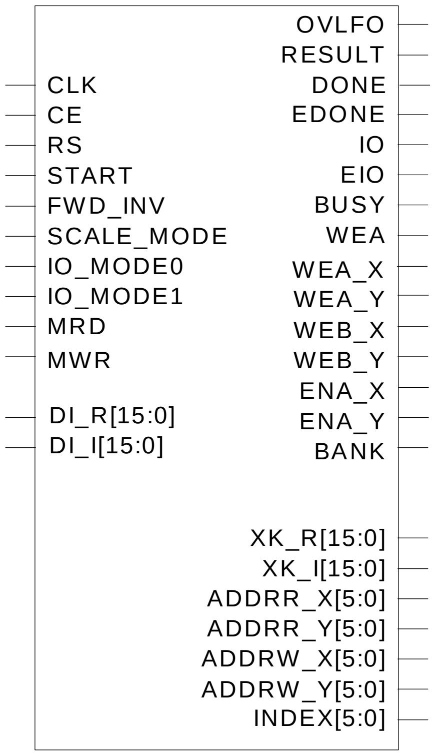

Figure 1: vfft64 symbol.

Table 1 defines the module pin functionality.

Signal Name	Direction	Description
CLK	Input	Master clock (active rising edge)
RS	Input	Reset (active high)
START	Input	Start processing (active high). This signal must be synchronized with CLK.
CE	Input	Clock enable (active high)
FWD_INV	Input	Defines if a forward (FWD_INV=1) or inverse (FWD_INV=0) FFT is performed. This signal is sampled when START is asserted for all memory configuration options (SMS, DMS and TMS). When the DMS and TMS configurations are selected, where FFTs are computed continuously, FWD_INV is sampled when MODE_CE is asserted by the core. This permits alternating forward and inverse FFTs to be performed.
SCALE_MODE	Input	FFT scaling control. When SCALE_MODE=0 the FFT output vector is scaled by 1/64. When SCALE_MODE=1 the FFT output vector is scaled by 1/128. This signal is sampled when START is asserted for all memory configuration options (SMS, DMS and TMS). When the DMS and TMS configurations are selected, where FFTs are computed continuously, SCALE_MODE is sampled when MODE_CE is asserted by the core. This permits different scaling options to be used for successive FFTs without any interruption to processing.
MWR	Input	Input data write strobe (active high). This signal must be synchronized with CLK.
MRD	Input	Result vector read strobe (active high). This signal must be synchronized with CLK.
DI_R[15:0]	Input	Input data bus – real component. The real component of the input data vector is presented to the core on this port. Two's complement data format is assumed.
DI_I[15:0]	Input	Input data bus –imaginary component. The imaginary component of the input data vector is presented to the core on this port. Two's complement data format is assumed.
IO_MODE0	Input	Together with the IO_MODE1 pin, these signals define the type of memory interface to be used, single, dual or triple memory space configurations. The memory configurations also effectively define the method by which data is presented to the core and read back from the core. The precise functionality of IO_MODE0 and IO_MODE1 are defined in the I/O Interface and Memory Configurations section of this document.
IO_MODE1	Input	I/O interface and memory configuration select pin – refer to the signal description for IO-MODE0 for details.
OVFLO	Output	Active high Arithmetic overflow indicator. Even when employing a 2-bit scale factor for each FFT processing phase, certain input signals can cause arithmetic overflow. This pin indicates that an internal arithmetic overflow has been generated. When additional scaling is employed by setting SCALE_MODE=1, there is no possibility of overflow occurring and this signal will not be active. OVFLO is removed when the core is reset by asserting RS, when START is asserted, or at the beginning of the next output result vector as indicated by DONE.
RESULT	Output	FFT result strobe (active high). This signal indicates that a new (I)FFT result vector is available. It frames the XK_R and XK_I output buses.
DONE	Output	FFT complete strobe (active high). DONE will transition high for one clock cycle at the end of the RESULT strobe.
EDONE	Output	Early done strobe (active high) EDONE goes high for one clock cycle immediately prior to RESULT going active.
IO	Output	IO cycle strobe. (active high) This signal is only intended to be used with the dual-memory-space core configuration.
EIO	Output	Early I/O strobe. (active high) This signal is only intended to be used with the dual-memory-space core configuration.
BUSY	Output	Core activity indicator (active high). This signal will go high in response to the start signal and will remain high while the core is processing data.
MODE_CE	Output	The operation mode signals FWD_INV and SCALE_MODE are sampled when

		MODE_CE is active. This is an active high signal that is asserted by the core for one clock period several clock cycles prior to the start of a new computation in the dual-memory-space and triple-memory-space core configurations.
WEA	Output	Block RAM port A write enable strobe (active high). This write signal is used for the single-memory-space configuration.
WEA_X	Output	Block RAM port A write enable strobe (active high). This write signal is used for the dual- and triple-memory-space configurations.
WEA_Y	Output	Block RAM port A write enable strobe (active high). This write signal is used for the dual and triple memory space configurations.
WEB_X	Output	Block RAM port B write enable strobe (active high). This write signal is used for the dual- and triple-memory-space configurations.
WEB_Y	Output	Block RAM port B write enable strobe (active high). This write signal is used for the dual and triple memory space configurations.
ENA_X	Output	Block RAM port A enable signal (active high). This memory enable signal is used for the single, dual and triple-memory- space configuration options.
ENA_Y	Output	Block RAM port A enable signal (active high). This memory enable signal is used for the triple-memory-space configuration option.
BANK	Output	Memory bank select signal to be used with the triple- memory-space core configuration.
ADDRR_X[7:0]	Output	Block RAM read address bus used for the single, dual and triple memory configurations.
ADDRR_Y[7:0]	Output	Block RAM read address bus used for the dual and triple memory configurations.
ADDRW_X[7:0]	Output	Block RAM write address bus used for the single, dual and triple memory configurations.
ADDRW_Y[7:0]	Output	Block RAM write address bus used for the dual and triple memory configurations.
INDEX[5:0]	Output	Output data re-ordering bus. This bus is used in the DMS and TMS memory configurations for re-ordering the transform result (XK_R and XK_I) as it is written to the result memory. The INDEX bus generates a radix-4 digit reversed sequence of addresses.
XK_R[15:0]	Output	FFT result bus – real component. The real component of the FFT result vector is presented on this bus. The values are in two's complement format.
XK_I[15:0]	Output	FFT result bus –imaginary component. The imaginary component of the FFT result vector is presented on this bus. The values are in two's complement format.

Table 1: 64-point FFT module pin definitions.

6 I/O Interface and Memory Configurations

To provide the system designer with maximum flexibility of the I/O interface and memory architecture for the vfft64 core, no data storage has been included in the core itself. The designer must supply the data memory as an external component(s). The core supplies a number of control signals that permit simple interfacing to one or several banks of on-chip Virtex Block RAM. These signals are not intended to be used for interfacing to Virtex distributed memory or memory devices external to the FPGA component. There are three basic memory configurations that are supported – 1. *single-memory-space (SMS),* 2*. dual-memory-space (DMS) (*also referred to as the *burst I/O* configuration) and 3. *triple-memory-space (TMS)* configuration. The *triple-memory-space* architecture is also referred to as the *ping-pong-memory* configuration. A memory configuration is selected using the *IO_MODE0* and *IO_MODE1* configuration pins as defined in Table 1. The memory interface employed also effectively defines the operation of the FFT I/O interface. Each memory space configuration is described in the following sections.

IO_MODE0	IO_MODE1	Memory Configuration
0	0	invalid
0	1	Single-memory-space (SMS)
1	0	Triple-memory-space (TMS)
1	1	Dual-memory-space (DMS)

Table 2: FFT Memory configuration mode selection.

6.1 Single-Memory-Space (SMS) Configuration

The *single-memory-space* configuration provides the simplest memory and I/O interface to the FFT core. An abstract model of this configuration is shown in Figure 2.

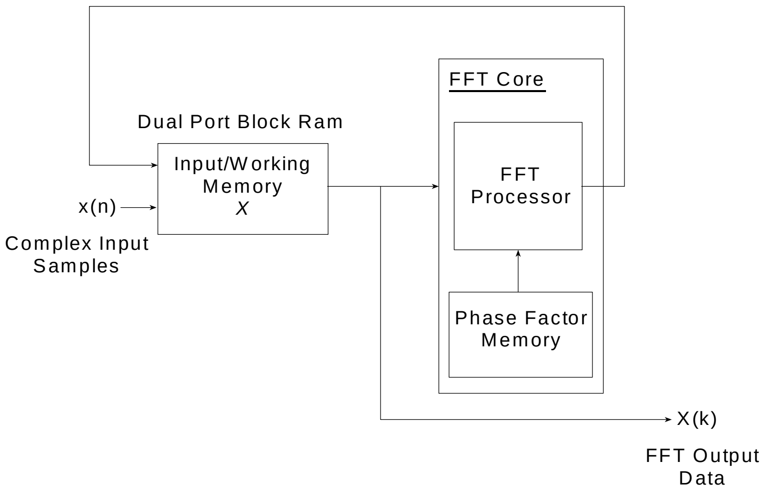

Figure 2: Abstract model of the *single-memory-space* FFT core interface.

A three-stage sequence of operations is used to compute transforms with this interface.

1. The input data vector is loaded into the input/working memory (labeled *X* in the figure). This processing step will be referred to as the *data load phase.*

2. When the input data load operation has completed the FFT engine is started – the *compute phase.*

3. When the FFT is complete, the result vector is read out of the *input/working memory X*. Note, that this is the same memory space that the original input data was supplied in. This processing step will be referred to as the *data-unload-phase*

One observation about this mode of operation is that the user experiences explicit I/O operations. While an input or output operation is in progress, the FFT core is idle. This inefficiency can be overcome (at the expense of additional memory banks) with the DMS and TMS configurations.

A detailed connection diagram is shown in Figure 3. The input data vector is a complex set of 64 16-bit precision samples. The real and imaginary components of the input samples, and the FFT result vector, are handled as two bus interfaces – a real and imaginary bus respectively. The complex memory bank, labeled *X* in the figure, consists of two 64 deep memories. A Virtex Block RAM may be configured as a 256x16 memory, so two Block RAMs are required to implement input/working/output memory bank.

In this figure the host input sample bus is designated *XN_[R/I][15:0]*.

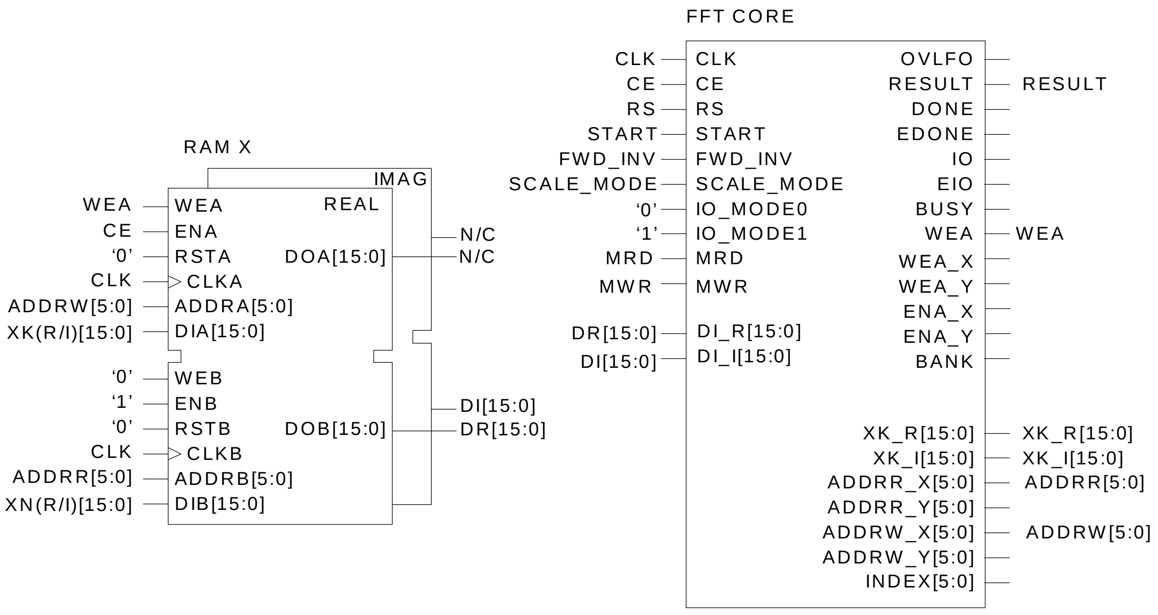

Figure 3: SMS core configuration – detailed connection diagram.

6.1.1 Data Load Phase

A data load operation is initiated by asserting the *MWR* signal for one clock period as shown in Figure 4. The host system supplies input samples on the XN_[R/I] buses on successive clock cycles. The data is written into memory on the rising edge of the clock. The *MWR* signal causes the FFT core to generate the memory address and control signals.

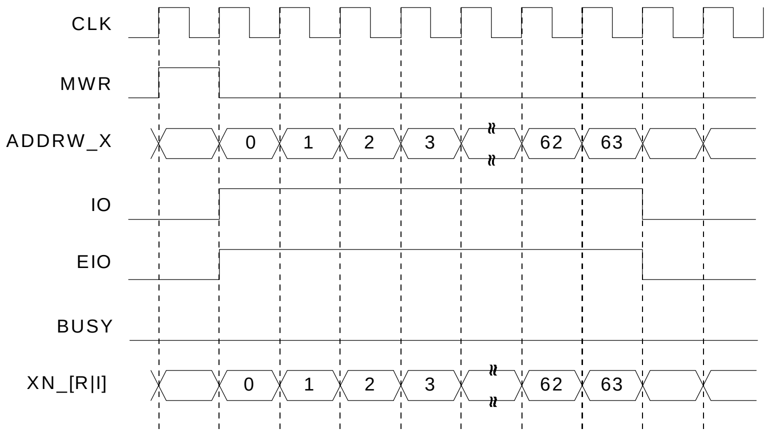

Figure 4: *Single-memory-space* configuration – input data load timing.

6.1.2 Compute Phase

Asserting *MWR* only initiates a data load operation, at the completion of the data load phase no further activity will ensue until the FFT engine is started by asserting the *START* signal. The timing for *START* is shown in Figure 5. For all of the modes, SMS, DMS and TMS, the operating mode input signals *FWD_INV* and *SCALE_MODE* are sampled on the rising edge of *CLK* with *START* used internally within the core as a qualifier (clock enable).

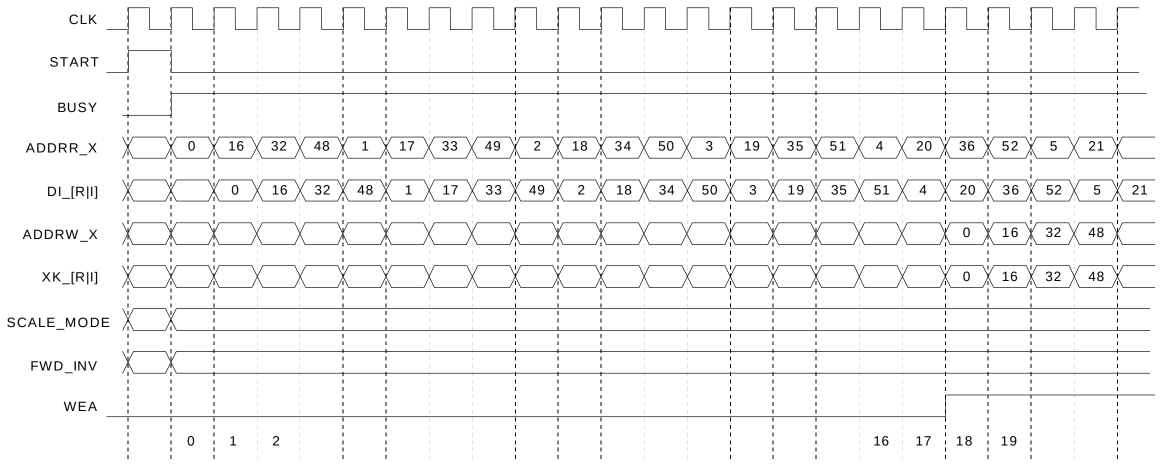

Figure 5: *Single-memory-space* configuration – FFT start timing.

The *FWD_INV* and *SCALE_MODE* function signals are sampled on the rising edge of *CLK* with *START* used as a qualifier.

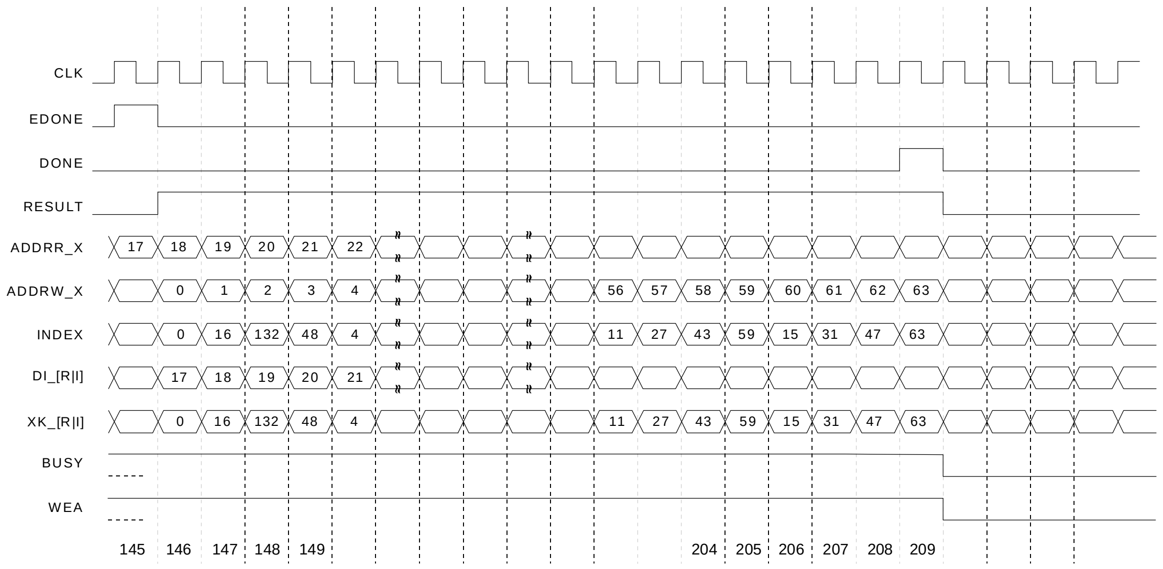

Figure 6: *Single-memory-space* configuration – result timing.

6.1.3 Result Unload Phase

Once the FFT is completed the result data is stored in the input memory bank – in fact the only memory bank in this configuration. The data is stored in *digit reversed order* [1]. The data is read back from memory by initiating a data read operation. The read operation unscrambles the data so that it is presented to the host system in natural order. A read operation is started by asserting *MRD* high for one clock period as shown in Figure 7. The FFT result vector is presented on Port B of the Block RAM.

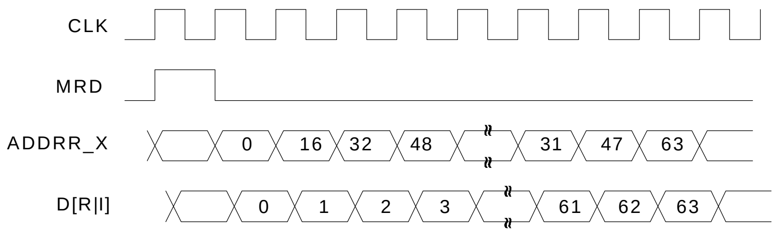

Figure 7: *Single-memory-space* configuration –result read timing.

6.1.4 Performance SMS Configuration

Three factors contribute to the FFT core performance in a system: 1. The input vector load time, 2. FFT computation time, and 3., the result unload time.

The input load and result unload operations each require 64 clock cycles. For example, using a 100 MHz system clock, each operation requires 0.64 microseconds.

The first FFT output sample is written to memory 146 clock cycles after *START* is asserted. The final sample is written to memory 209 clock cycles following *START*. The computation phase alone is 1.92 microseconds in duration. Using an 80 MHz clock the execution time is 2.6 microseconds.

6.2 Dual-Memory-Space (DMS) Configuration

The dual memory configuration shown in Figure 8 allows input, computation and output operations to be overlapped, so that the FFT core is never left in an idle state waiting for a host I/O operation. To understand how this mode works recall that an FFT is computed by making multiple processing passes over the input data. For the case of a 64-point FFT, three such radix-4 computation stages are required. During the final processing phase in the DMS configuration, instead of returning the processed data to the input/working buffer *X,* it is stored in the output memory bank *Y.* Concurrent with this operation, a new vector of input data is written into memory bank *X*. Therefore, data load, computation and output operations are overlapped, and no computation cycles are wasted due to I/O cycles. The new data must be *burst* into memory because only 64 clock cycles are available in which to perform the operation. The burst I/O operation must be synchronized with the FFT engine. Several control signals are generated by the core to facilitate this operation. When the FFT output vector has been placed in the output buffer, the host system can use the second port of the dual port Block RAM to read back the result and pass this data to down-stream components in the system.

Because the FFT is effectively being performed out-of-place (in contrast to the in-place process of the SMS configuration) the result vector can be written to the output buffer memory *Y* in natural order. An address indexing bus (*INDEX*) is supplied for this purpose. The host system should therefore treat the output samples in memory space *Y* as a naturally ordered FFT result vector.

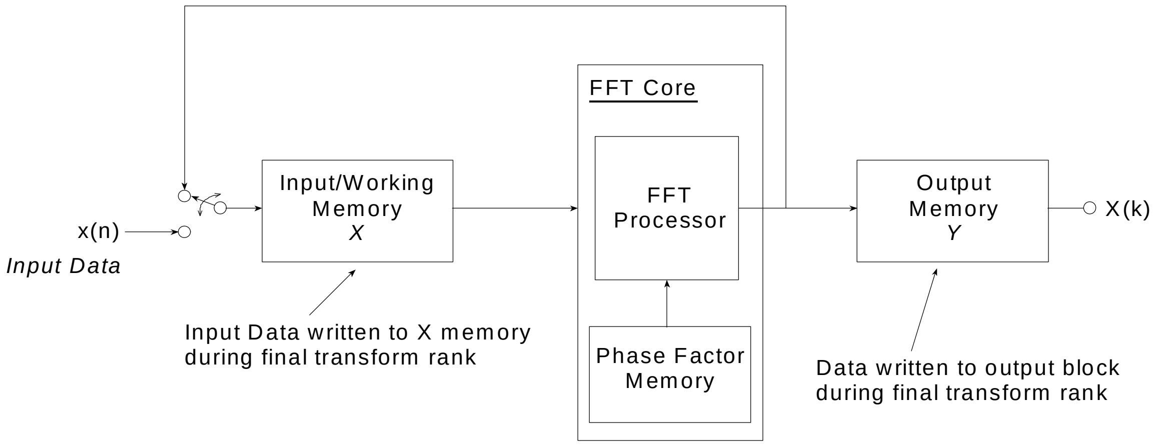
FFT Core
Input/Working Memory X
FFT Processor
Output Memory Y
X(k)
x(n)
Input Data
Phase Factor Memory
Input Data written to X memory during final transform rank
Data written to output block during final transform rank

Figure 8: *Dual-memory-space* FFT core interface.

The detailed schematic for the DMS FFT configuration is shown in Figure 9. The host system must supply a multiplexer to share port A of the dual port Block RAM between the input samples *XN_[R/I]* and the FFT result bus *XK_[R/I].* A multiplexer control signal (*IO)* is generated by the core to make this interface simple to implement. Even though only one port of the memory is used for writing, due to the concurrent read and write operations, a dual port memory must be used. Because of certain timing events in the core, it is not possible to use the port B input databus – it should be tied to a suitable inactive state.

The *RESULT* strobe indicates that a new result vector is appearing on the *XK_[R/I]* buses. The strobe exactly frames the vector and can be used as a write enable signal for memory bank *Y.* The FFT samples appearing on the *XK_[R/I]* bus are presented in digit (radix-4) permuted order. The *INDEX* bus can be used to perform data re-ordering as the results a written to memory.

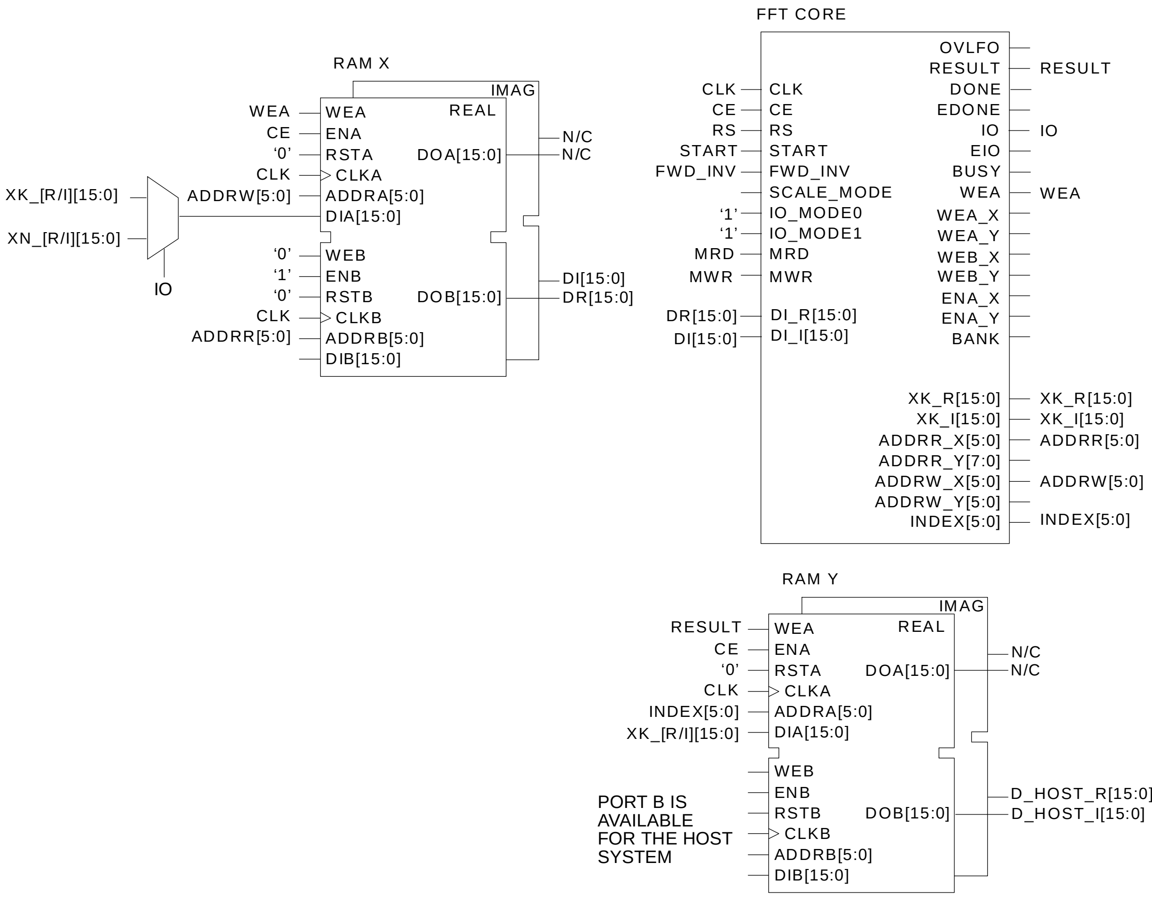

Figure 9: DMS core configuration – detailed connection diagram.

This mode of operation would typically be used by first performing a data load operation – exactly like the memory write operation described for the SMS configuration. After the initial data load has completed, the FFT engine is started by asserting *START* synchronously with *CLK. START* is only employed in this mode for starting the very first transform of a sequence of FFTs – ***IT***

SHOULD NEVER BE ASSERTED AGAIN UNLESS THE INTENTION IS TO RE-START THE ENTIRE PROCESSING ENGINE.

The *FWD_INV* and *SCALE_MODE* pins are sampled when *START* is applied. Once the core has entered its computation phase, and FFTs are being performed back-to-back, *FWD_INV* and *SCALE_MODE* will be sampled when *MODE_CE* is asserted.

MRD would typically be tied inactive (low) when using the DMS memory configuration.

Once started, the core will continue processing input vectors in an uninterrupted manner. This does not imply that data can be continuously streamed into memory, memory write operations must be synchronized with the core using the *IO* and/or the *EIO* signals. Observe that there are the I/O operations and processing are completely overlapped, unlike the SMS FFT architecture.

Figure 10 provides timing information for the output result and I/O operations. The *IO* signal exactly frames a user data load operation. *EIO* is an *early* I/O strobe that is provided as a pre-emptive signal to indicate to the data source that a load operation is about to be initiated.

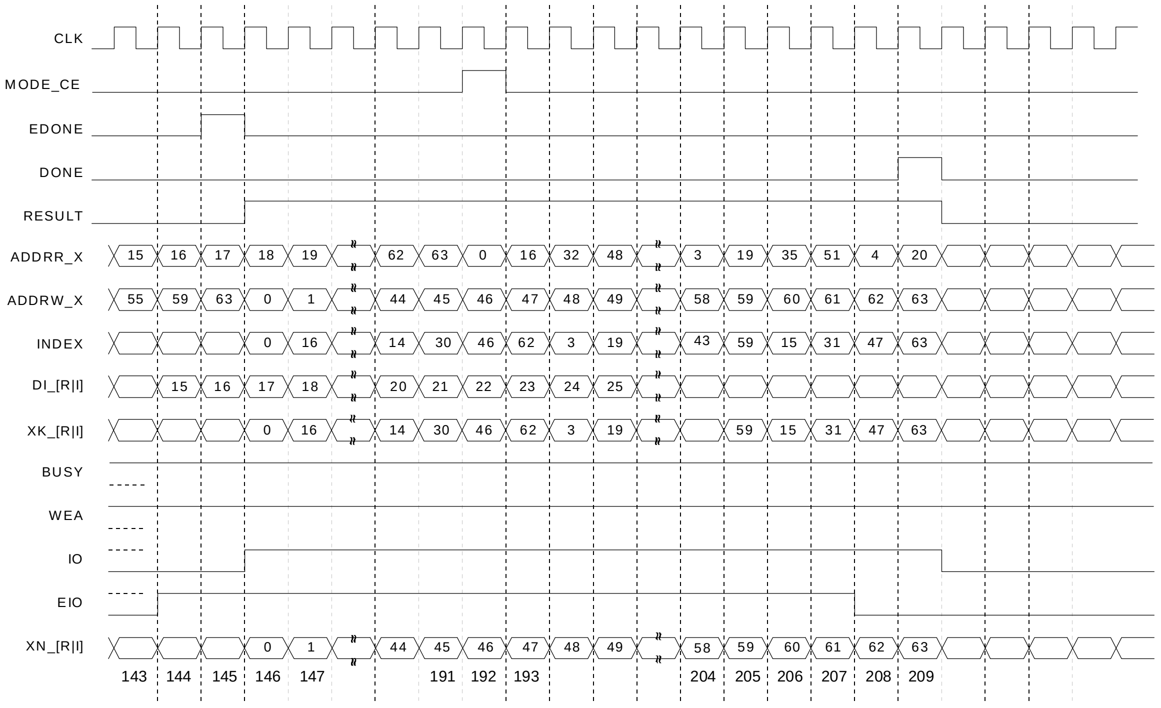

Figure 10: *Dual-memory-space* configuration – result and I/O timing.

Also shown is the timing for the *MODE_CE* signal. This signal indicates when the operation mode pins *FWD_INV* and *SCALE_MODE* are sampled. Using this signal, alternating forward and inverse FFTs can be computed back-to-back in a seamless fashion. In addition, the scaling mode may be changed between successive computations. The *FWD_INV* and *SCALE_MODE* signals need only be valid during the single clock period indicated by *MODE_CE*. At other times the system is free to change the state of these signals – of course this will have no impact on the processing currently in progress.

6.2.1 Performance DMS Configuration

After the initial memory load operation a new FFT result vector will be available every 192 clock cycles **after the first FFT has completed**. As measured from the assertion of *start*, the very first FFT requires 209 clock cycles to complete. The additional 17 clock cycles required by the first FFT are associated with the pipeline depth of the FFT arithmetic unit.

For a 100 MHz, the 192 cycle count equates to a transform time of 1.92 microseconds.

6.3 Triple-Memory-Space (TMS) Configuration

While the DMS configuration ensures that the FFT core is supplied with data 100% of time so that no potential computation cycles are wasted, it requires the user to burst data into the input memory bank synchronously with the core operation. The I/O requirements can be relaxed with the TMS configuration. This model of computation is shown conceptually in Figure 11. There are two input/working memory banks, designated *X* and *Y* respectively, and one output buffer *Z*. While the FFT core is working with one memory bank, say *X*, the host system can load data into the alternate memory bank, *Y* in this case. The result vector is always written to buffer *Z*. The host system has complete access to the input memory that is not currently involved in the calculation, and so has more flexibility, compared to the DMS configuration, over the way in which a new input vector is written to memory. For the case of a 64-point FFT an average of 3 clock cycles are available for writing each new input sample to memory. When the FFT core has finished processing the current input data, the memory banks are swapped and the data load and computation continues on the alternate memory buffers. This type of processing is often referred to as a *ping-pong* mode of memory architecture.

The output result is written to bank *Z* in much the same manner as for the DMS configuration. Again, the FFT output samples are presented on the result bus in digit-reversed order and can be re-ordered as they are written to memory. The FFT core can use one port of dual-port RAM for writing and the host can use the other port for unloading the result once the write operation has completed.

As with the DMS configuration I/O operations are completely overlapped with the FFT computation. However, unlike the that mode, the new input samples can be presented to the input buffer in a continuous, rather than as a burst operation. A new sample is written to memory every 3 clock cycles. All of the memory interface signals are generated by the core to simplify system integration.

It may be tempting to start the unload operation prior to completion of the vector write operation since two independent memory interfaces are available on the output buffer. However, remember that the results are not written to memory in linear order but in digit-reversed order, and so it is recommended that the host system stall any memory activities until the result write operation has completed.

The *FWD_INV* and *SCALE_MODE* pins are sampled when *START* is applied. Once the core has entered its computation phase, and FFTs are being performed back-to-back, *FWD_INV* and *SCALE_MODE* will be sampled when *MODE_CE* is asserted.

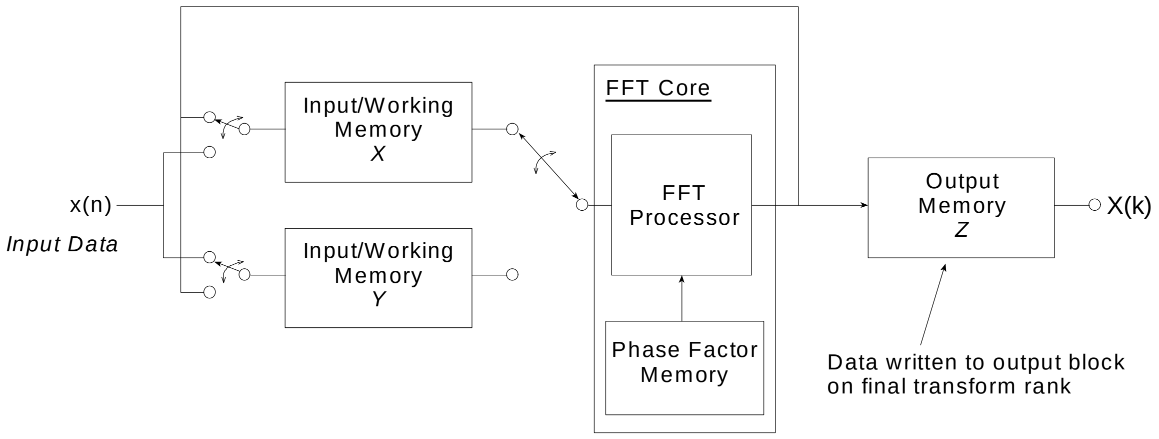

Figure 11: Abstract model of the *triple-memory-space* FFT core interface.

A detailed interconnection diagram for the TMS configuration is provided in Figure 12. In addition to the three complex 64 x 16 memory buffers (3 Virtex Block RAMs in total), the design requires a simple 2-to-1 16-bit complex multiplexer. To simplify the timing of the interface the FFT core provides a multiplexer control signal called *BANK*. This architecture can support a continuous stream of samples, a new sample being presented to the system every 3 clock cycles.

This mode of operation would typically be used by first loading memory bank *X* using a similar procedure for the data load operation in the SMS configuration. The timing for the memory write operation is shown in Figure 13. The FFT engine would then be activated by asserting *START* for a single period of *CLK* as shown in Figure 14. This figure also illustrates the timing for *BANK* and the memory interface signals *WEA_X, WEA_Y, WEB_X, WEB_Y, ENA_X* and *ENA_Y*.

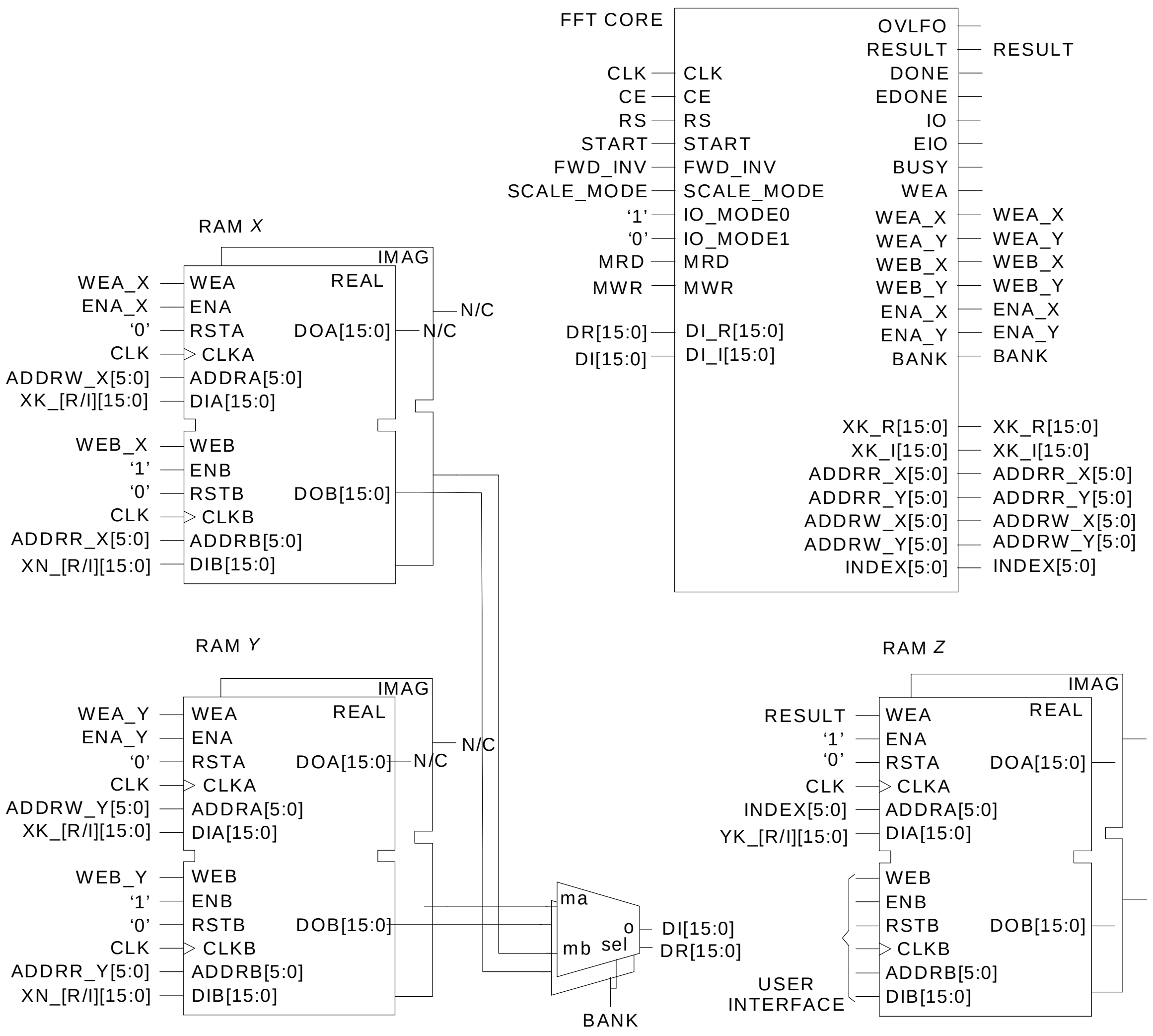

Figure 12: TMS core configuration – detailed connection diagram.

The *MODE_CE* output signal provides the same functionality in this configuration as in the DMS mode of operation. Using *MODE_CE*, the host system may perform alternating forward and inverse FFTs, in addition to changing how scaling is handled between successive input vectors.

MRD would typically be tied inactive (low) when using the TMS memory configuration.

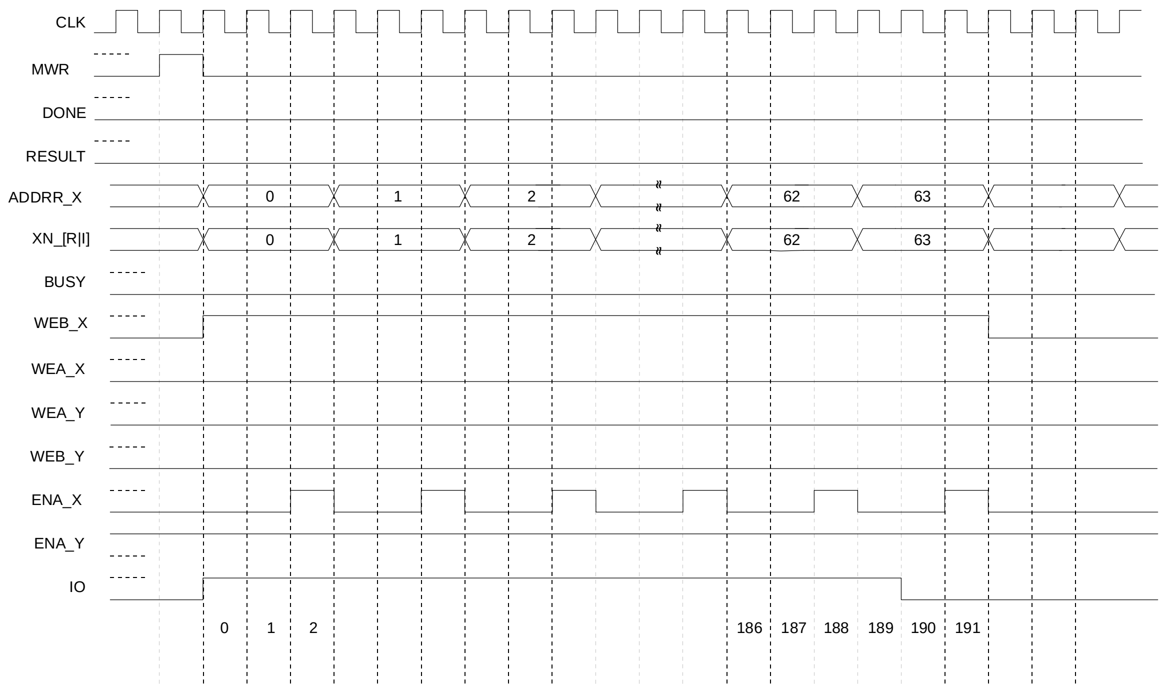

Figure 13: TMS core configuration – memory write operation.

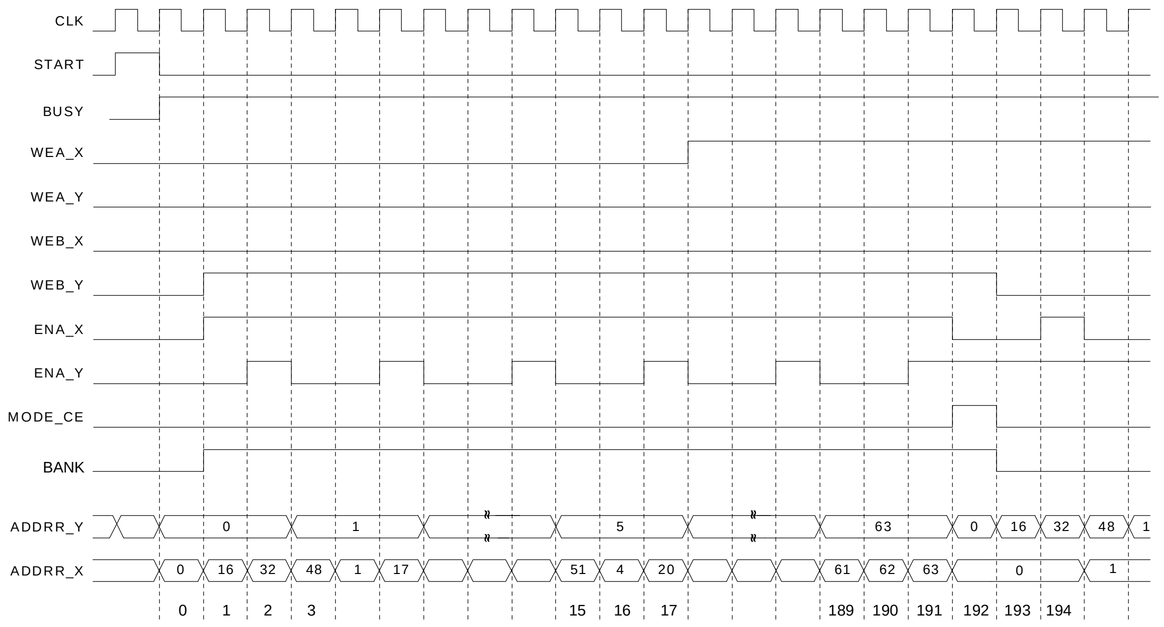

Figure 14: TMS core configuration – FFT start timing.

Figure 15 shows the FFT output timing. The address bus *INDEX* can be used to unscramble the data as it appears on the core result bus *XK_[R/I]*. The *RESULT* strobe indicates when valid data is available on the FFT output bus.

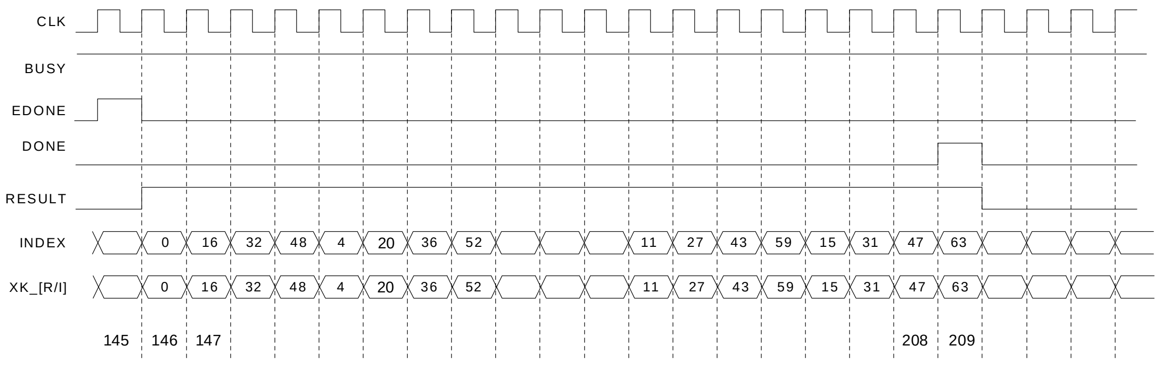

Figure 15: TMS core configuration – result and *INDEX* address bus timing.

6.3.1 Performance TMS Configuration

After the initial memory load operation a new transform result is available every 192 clock cycles.

7 Core Resource Utilization

The 64-point FFT Core occupies 1161 logic slices. The geometry of the RPM requires it to be placed in a XCV300 or larger device.

8 Behavioral Simulation

Release Version 1.0 of the vfft64 core has VHDL behavioral model but does not include a verilog behavioral model.

9 Implementation

The vfft64 core is supplied as a group of edif netlists. The top level netlist is called *vfft64.edn*. All of the netlists that are delivered with the core must be present in the user’s project directory. The edif netlist files are:

```
vfft64.edn
xdsp_cnt9.edn
```

```
xdsp_cos64.edn
xdsp_mul16x17z4.edn
xdsp_mux2w1.edn
xdsp_mux2w16.edn
xdsp_mux2w4.edn
xdsp_mux3w1.edn
xdsp_mux4w16.edn
xdsp_radd16.edn
xdsp_radd17.edn
xdsp_ramd16a4.edn
xdsp_reg15.edn
xdsp_reg16.edn
xdsp_reg16b.edn
xdsp_reg16l.edn
xdsp_rsub16.edn
xdsp_rsub16b.edn
xdsp_rsub17.edn
xdsp_rsub17b.edn
xdsp_sin64.edn
xdsp_tcompw16.edn
xdsp_tcompw16b.edn
xdsp_tcompw17.edn
```

10 Ordering Information

This macro comes free with the Xilinx CORE Generator. For additional information contact your local Xilinx sales representative, or e-mail requests to dsp@xilinx.com.

11 References

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

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

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