49 FFT Accelerator (FFTA)

The FFT Accelerator (FFTA) performs memory to memory FFT/IFFT operations without core software intervention. Additionally, the FFTA architecture allows execution of complex, pipelined, memory to memory algorithms including ping-ponged, windowed frequency domain filtering and very large FFTs. The FFTA may also be used in conjunction with minimal computation support from a core in applications such as the overlap-add operations required for large frequency domain based convolutions.

FFTA Features

The following list describes the FFTA features

- Supports both complex and real FFT and IFFT operations
- Supports 64, 128, 256, 512, 1024, 2048 points in small FFT mode and 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304 points in large FFT mode.
- Supports the IEEE-754/854 single-precision floating-point data format, round to even
- Radix-4 butterfly efficiency at a radix-2 (integer power of two) point granularity
- Automatic insertion of zeros for real FFTs
- Supports automatic conjugating of the twiddle factors for IFFT
- Supports automatic scaling of FFT and IFFT inputs
- Hardware support for windowing and frequency domain filtering
- Hardware support for magnitude squared FFT output
- Hardware support for pipelined data flow
- Dedicated high speed DMA engines for data load and dump with a data width 64-bit clocked by *SYSCLK*
- Supports data and coefficient access from both on-chip (L1/L2) and off-chip memories (L3)
- Optional support for bypassing the compute engine to perform high speed memory to memory MDMA transfers
- Clock division options for power reductions, supports 1:1, 1:2, 1:4 and 1:8 clock ratio modes

FFTA Functional Description

The FFTA module provides the following functionality.

Complex and real FFT and IFFT operations

The inputs and outputs for the complex FFT routines are packed arrays of floating point numbers. In a packed array the real and imaginary parts of each complex number are placed in alternate neighboring elements.

Hardware support for windowing, frequency domain filtering and pipelined data flow

Windowing is a technique used to shape the time portion of your measurement data, to minimize edge effects that result in spectral leakage in the FFT spectrum. By using Window Functions correctly, the spectral resolution of your frequency-domain result will increase. Data pipe-lining helps to process one set of frame while other is being DMAed into and out of the accelerator.
Dedicated high speed DMA engines for data load and dump with a data width 64-bit clocked by SYSCLK This enables faster data movement between the memory internal to the accelerator and memories external to the accerator (L1/L2/L3).

Supports data and coefficient access from both on-chip (L1/L2) and off-chip memories (L3)

This helps to be able to process larger data buffers which can't be fit in to the limited size of the L1/L2 memories.

ADSP-SC58x FFTA Register List

High-Speed FFT Compute Unit

Table 1: ADSP-SC58x FFTA Register List

Name	Description
`FFTA_CTL`	Control Register
`FFTA_INST[nn]`	Instruction Memory
`FFTA_LC[nn]`	Loop Counter Value Register
`FFTA_PC`	Program Counter Register
`FFTA_SCALE`	FFT/IFFT Scale Factor
`FFTA_STAT`	Status Register
`FFTA_THREADOFFSET`	Thread Count Offset Register
`FFTA_WCTL`	Wrapper Control Register

Table 1: ADSP-SC58x FFTA Register List (Continued)

Name	Description
FFTA_XFRLEFT[nn]	Load/Dump Transfer Left Register

FFTA Definitions

To make the best use of the FFTA, it is useful to understand the following terms.

Small FFT

Small FFT corresponds to the FFT/IFFT operation where number of points are less than or equal to 2048. These operations can be accomplished directly with the local memory supported by the FFTA.

Large FFT

Large FFT corresponds to the FFT/IFFT operation where the number of points is greater than 2048. These operations are not directly supported by the FFTA because of the limited local memory. These are carried out in multiple stages using a divide and conquer approach by performing a number of small FFT operations.

CCES

CrossCore Embedded Studio

RTL

CrossCore Embedded Studio C/C++ Runtime Library

CCES RTL Manual

CrossCore Embedded Studio C/C++ Compiler and Library Manual for SHARC® Processors

FFTA Block Diagram

The FFTA contains a compute engine that operates in conjunction with a High Speed Distributed DMA Engine (HS-DDE). The compute engine appears as a data sink and data source to the two dedicated DDEs. The DDEs stream data into and out of the engine's FIFOs. To support faster data movement, the HS-DDE engine has a bus width of 64 bits and runs at SYSCLK speed. The compute engine data bus is 256 bits wide. The data transfer between the DMA engines and compute engine happens via data pack and unpack logic.

The FFTA compute engine runs at *FFTCLK*. The value of *FFTCLK* can go up to a maximum of *SYSCLK*. However, to achieve lower power consumption, it is also possible to operate the compute engine at frequencies lower than *SYSCLK*. This can be done by programming the *FFTCLK*: *SYSCLK* frequency ratio in the FFTA_WCTL register.

The two channel high speed DMA engines can also (optionally) be used for high speed memory to memory DMA (MDMA) transfer if the FFT compute engine is not used in the system.

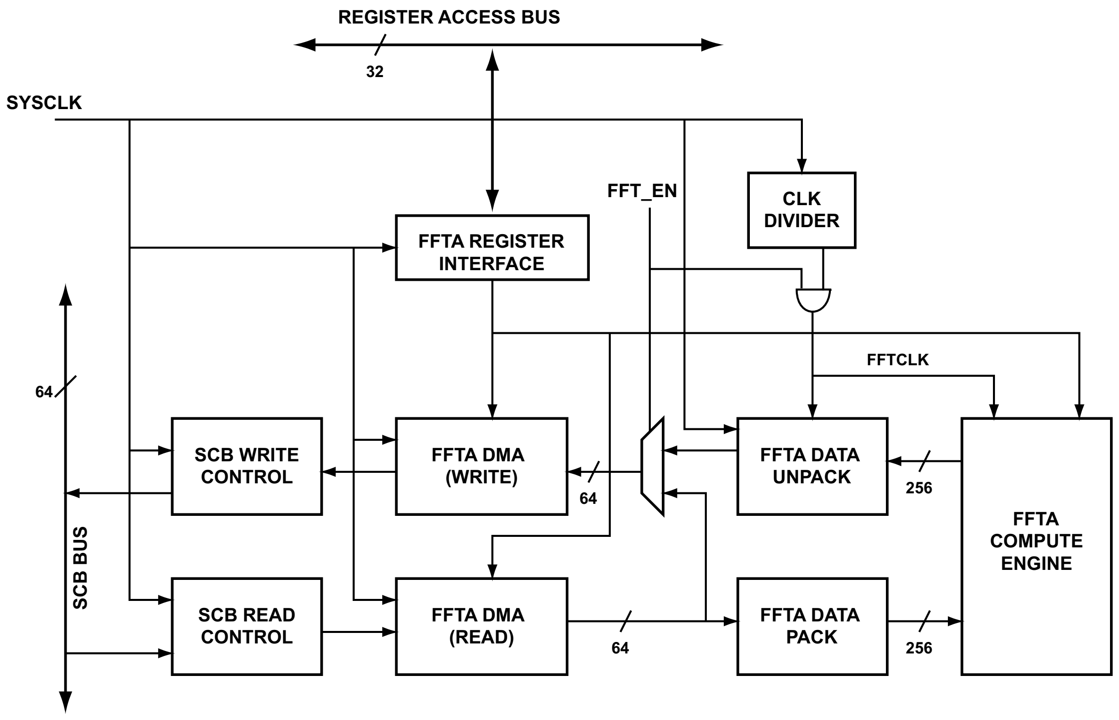

Figure 1: FFTA Block Diagram

FFTA Programming Model

A software library containing support for using the FFTA is provided with the CrossCore Embedded Studio. Using this library, C or C++ programs running on an ARM Cortex A5 or SHARC ADSP-215xx and ADSP-SC5xx core can access the FFTA to implement various specific use scenarios as discussed in the following sections. Analog Devices Inc. does not support programming the FFTA at register level (except for the optional programming of the `FFTA_WCTL` register). However, detailed description of the FFTA registers is provided in this chapter for debugging purpose.

Use Cases Supported by FFTA

The following section provides the conceptual description of the FFTA use cases supported by the CCES RTL APIs. For exact description on the programming model for using the CCES APIs, please refer to the CCES Run Time Library manual.

1. Single-Shot FFT
2. Pipelined small FFT
3. Pipelined small interleaved FFT and IFFT operations

Single-Shot FFT

This use case is equivalent to the case where the core calls the CCES RTL's FFT/IFFT core library function. An API is called to perform an FFT/IFFT operation on a set of data and wait for the FFT/IFFT processing to finish. In this case all three units of the FFTA (Input, Compute, and Output) operate in a sequential manner as illustrated by the **Single-Shot** figure. Notice how each frame passes through the stages of the FFTA block (input > compute > output) sequentially. The total time required to complete the FFT operation is equal to the time taken to load the input data plus time taken to compute the FFT/IFFT plus the time taken to dump the output data. CCES provides an FFTA version of these functions/APIs and the APIs supporting this use case always operate in synchronous mode.

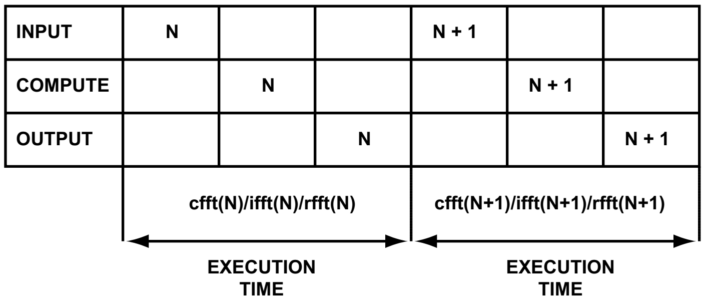

Figure 2: Single-Shot

The APIs supporting this use case are further classified into two categories:

1. **Drop-in replacement RTL Functions.** These are the FFTA versions of the core RTL functions cfftN, ifftN, rfftN (for example cfft64/ifft128/rfft256), cfft, ifft, rfft. For more details on the usage of these functions, refer to the CCES RTL Manual.
2. **FFTA Specific RTL Functions.** These are few additional RTL functions which support the following advantages over the drop-in replacement functions:
   - Separate functions for small and large FFTs with lesser memory footprint than the drop-in replacement functions cfft, ifft, and rfft (which include code to handle both small and large FFTs).
   - Additional set of functions with the support for windowing and scaling. Windowing allows pointwise multiplication of input data with a window function (for example Hanning window) before performing the FFT operation. Scaling allows multiplying all the input data points with a constant scaling factor.

Pipelined Small FFT

The piplined small FFT is used where an FFT/IFFT operation with a fixed number of points is performed continuously on more than one set of input data. The FFTA architecture supports pipelining of the input and output data which allows the FFTA to load one set of input data and dump another set of processed output data in parallel while another set of data (fetched in the previous pipeline cycle) is being processed. This helps to suppress the input and output DMA overheads while the FFTA compute engine is busy processing the data. This way, programs can take maximum advantage of the FFT compute engine's performance.

To support such cases, the FFTA APIs provide a mechanism where the FFTA is configured one time to perform the FFT/IFFT operation with a fixed number of points in a continuous pipelined manner. Next, the APIs related to the data transfers (asynchronous) are called to send the input data to and collect the output data from the FFTA. After all the data is processed, the FFTA can be closed. For more details on these APIs, refer to the CCES RTL manual.

The **Pipelined Small FFT** figure illustrates this use case. There is large startup latency (Input DMA time + Compute time + Output DMA time) involved to fill the pipeline with three frames. After that, each frame can be processed in a lesser steady state time (maximum of Input DMA time, Compute time, and Output DMA time).

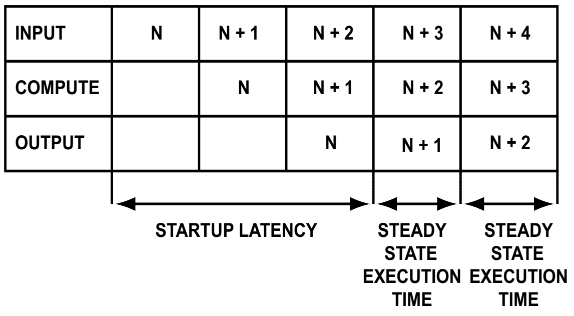

Figure 3: Pipelined Small FFT

Pipelined Small Interleaved FFT and IFFT Operations

Pipelined small interleaved FFT and IFFT operation are useful when data is first converted to the frequency domain, processed in the frequency domain and converted back to the time domain. The **Typical Data Flow Frequency Domain Processing** figure illustrates this where a particular input data frame N is first converted to the frequency domain (N') with an FFT operation. This FFT output is then processed in the frequency domain to produce the output frame N". The processed output frame (N") is then converted in to time domain output frame (N"').

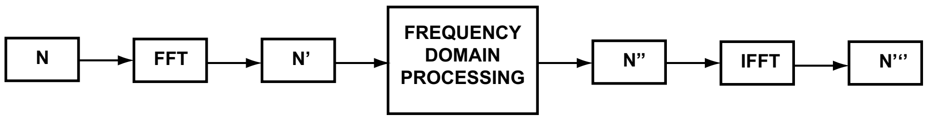

Figure 4: Typical Data Flow Frequency Domain Processing

The FFTA can be used to perform the above operation on multiple sets of data in a pipelined manner as shown in the **Pipelined Interleaved FFT/IFFT** figure. Unlike in the "Pipelined Small FFT" section, the compute engine performs both FFT and IFFT operations together in an interleaved manner. Note that the input data for the first valid IFFT operation (1") is available only when the FFT output of the first frame

(1’) is processed by the core. It is necessary to send dummy input data at the start of the pipeline for the initial few IFFT operations as shown in the **Pipelined Interleaved FFT/IFFT** figure. The output of these IFFT operations can be ignored. Similarly, dummy data must be sent at the end of the pipeline for the last few FFT operations. The output of these FFT operations can be ignored.

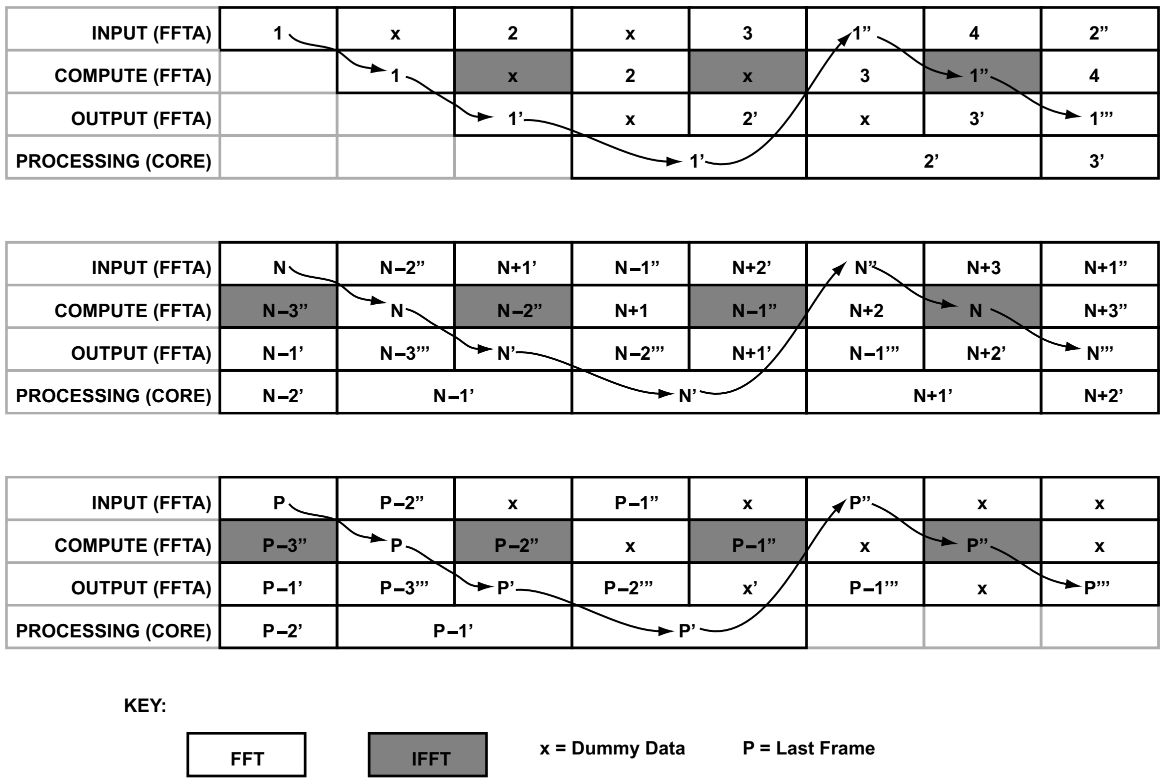

Figure 5: Pipelined Interleaved FFT/IFFT

Similar to the "Pipelined Small FFT" section, the FFTA APIs provide a mechanism where the FFTA can be configured once to perform interleaved FFT and IFFT operations with a fixed number of points in a continuous pipelined manner. After that, programs just need to call the APIs related only to the data transfers (asynchronous) to send the input data to and collect the output data from the FFTA. After all the data is processed, the FFTA can be closed. For more details on these APIs, refer to the CCES RTL manual.

ADSP-SC58x FFTA Register Descriptions

High-Speed FFT Compute Unit (FFTA) contains the following registers.

Table 2: ADSP-SC58x FFTA Register List

Name	Description
FFTA_CTL	Control Register
FFTA_INST[nn]	Instruction Memory
FFTA_LC[nn]	Loop Counter Value Register
FFTA_PC	Program Counter Register
FFTA_SCALE	FFT/IFFT Scale Factor
FFTA_STAT	Status Register
FFTA_THREADOFFSET	Thread Count Offset Register
FFTA_WCTL	Wrapper Control Register
FFTA_XFRLEFT[nn]	Load/Dump Transfer Left Register

Control Register

The FFTA_CTL register is used to reset, start, pause and single step the HPFP-FFT engine.

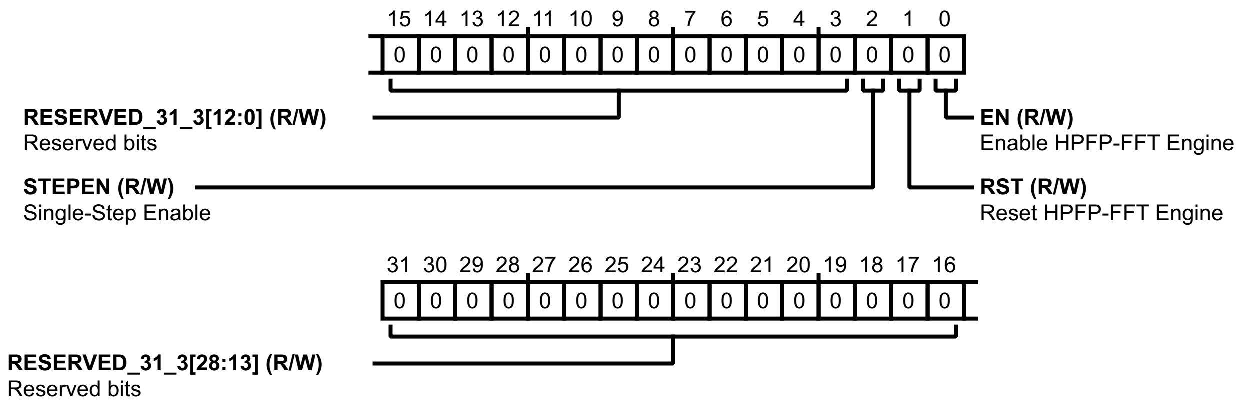

Figure 6: FFTA_CTL Register Diagram

Table 3: FFTA_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/W)	RESERVED_31_3	Reserved bits.

Table 3: FFTA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	STEPEN	Single-Step Enable. When the FFTA_CTL.STEPEN and FFTA_CTL.EN bits are set together the engine to release the instruction currently pointed to by the PC to execution, increment the PC then stop execution. The FFTA_CTL.STEPEN and FFTA_CTL.EN bits are automatically cleared when the released instruction completes execution. The FFTA_CTL.EN bit or FFTA_CTL.STEPEN bit can then be read to determine if the instruction step has completed. Note that a single step advances the PC one increment and at least one thread completes execution but one or two other threads may still not have completed execution. The FFTA_CTL.STEPEN control bit is self-clearing.	
		0	Single-Step Disabled
		1	Single-Step Enabled
1 (R/W)	RST	Reset HPFP-FFT Engine. When the FFTA_CTL.RST bit is set all MMRs are cleared. The FFTA_CTL.RST bit is self-clearing.	
		0	Not reset HPFP-FFT engine
		1	Reset HPFP-FFT engine
0 (R/W)	EN	Enable HPFP-FFT Engine. The FFTA_CTL.EN bit must be set for the engine to process instructions. When the FFTA_CTL.EN bit is cleared the engine is in an idle state.	
		0	Disable HPFP-FFT engine
		1	Enable HPFP-FFT engine

Instruction Memory

The FFTA_INST[nn] MMR registers hold (up to 64) FFT instructions.

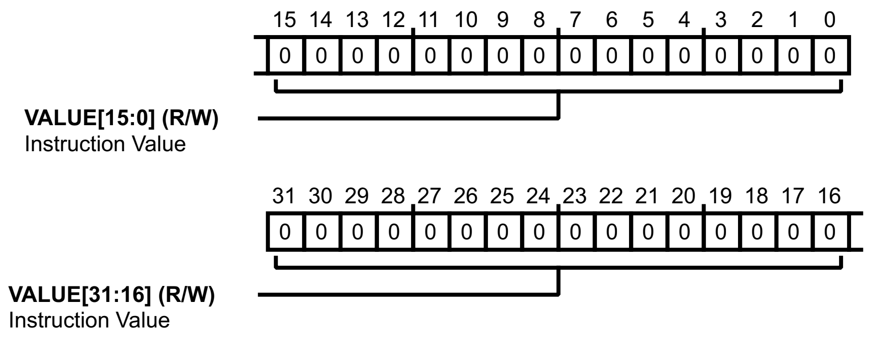

Figure 7: FFTA_INST[nn] Register Diagram

Table 4: FFTA_INST[nn] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Instruction Value. The FFTA_INST[nn].VALUE bit field contains 64 instruction MMR to hold FFT instructions.

Loop Counter Value Register

The FFTA_LC[nn] register holds the current value of the corresponding loop counter.

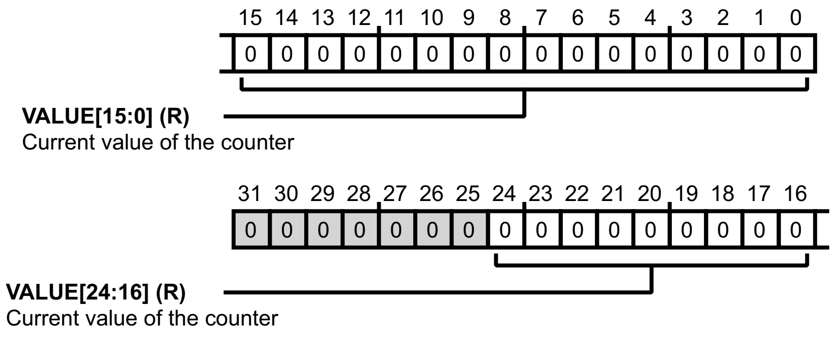

Figure 8: FFTA_LC[nn] Register Diagram

Table 5: FFTA_LC[nn] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:0 (R/NW)	VALUE	Current value of the counter.

Program Counter Register

The FFTA_PC register contains the MMR address offset of the current instruction in the instruction queue. The PC advances unless the instruction it points to is not acknowledged. When the instruction pointed to is acknowledged it starts the task and immediately advances. See the rules for instruction acknowledge in the Sequencer section of the Programming reference.

Note that the nop(), jumpCNZ(), load_loop_cntr(), incr_thread_offset() and load_scale() instructions are immediately acknowledged. Also note that only 6 bits are used, which implies that this counter is incremented in the fashion of modulo 64. For example, If the current PC=63, after it is incremented by 1, PC=0.

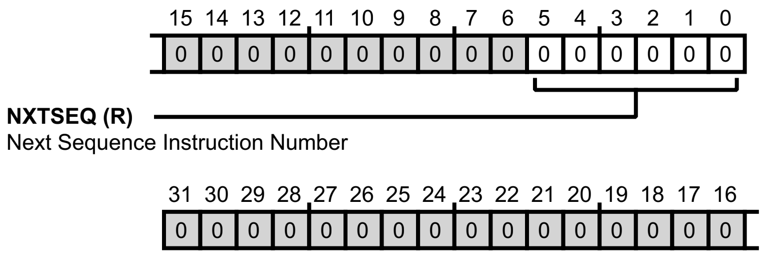

Figure 9: FFTA_PC Register Diagram

Table 6: FFTA_PC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/NW)	NXTSEQ	Next Sequence Instruction Number. The FFTA_PC.NXTSEQ bits provide the value by which the PC advances unless the instruction it points to is not acknowledged. When the instruction pointed to is acknowledged it starts the task and immediately advances. See the rules for instruction acknowledgement in the Sequencer section. The nop(), jumpCNZ(), load_loop_cntr(), incr_thread_offset() and load_scale() instructions are immediately acknowledged. Note that only 6 bits are used, which implies that this counter is incremented in the fashion of modulo 64. For example, If the current PC=63, after it is incremented by 1, PC=0.

FFT/IFFT Scale Factor

The FFTA_SCALE register is loaded using the load_scale() instruction. The scale factor is in a single-precision IEEE format. On reset the SCALE is set to 1.0 (floating point).

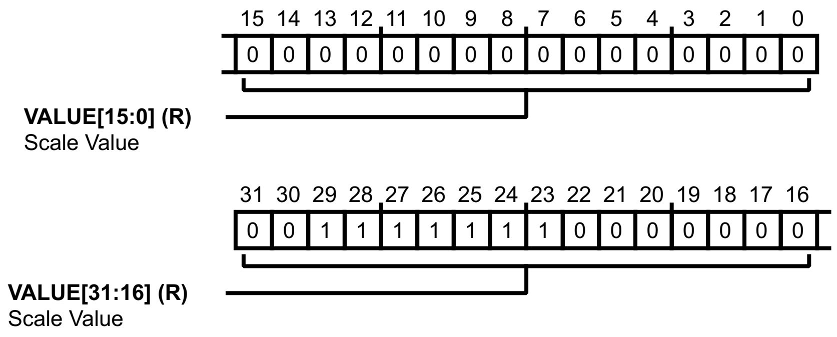

Figure 10: FFTA_SCALE Register Diagram

Table 7: FFTA_SCALE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Scale Value. Reset to 1.0

Status Register

The `FFTA_STAT` register indicates the status of FFT operations. Sticky status flags remain set until cleared by reset. In the status description the term math operation or math result refers to all multiply, add and subtract results including intermediate operations.

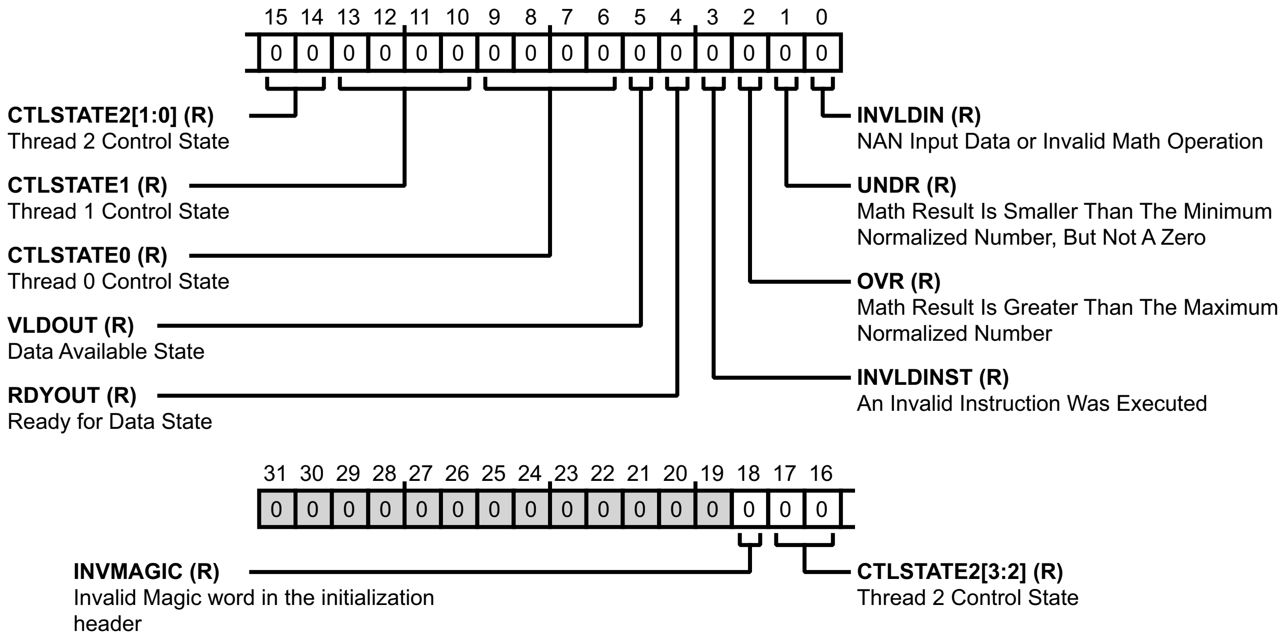

Figure 11: FFTA_STAT Register Diagram

Table 8: FFTA_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	INVMAGIC	Invalid Magic word in the initialization header. Invalid magic word in the initialization header.	
		0	Magic word is ok
		1	Magic word is not ok

Table 8: FFTA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:14 (R/NW)	CTLSTATE2	Thread 2 Control State. The `FFTA_STAT.CTLSTATE2` bit indicates the data available state (reflects state of fft. valid_o).	
		0	idle state
		1	load coef state
		2	load data state
		3	dump data state
		4	fft state
		5	multiply state
		6	maganitude squared state
		7	generate twiddles state
		8	bit reversal state
13:10 (R/NW)	CTLSTATE1	Thread 1 Control State. The `FFTA_STAT.CTLSTATE1` bit indicates the data available state (reflects state of fft. valid_o).	
		0	idle state
		1	load coef state
		2	load data state
		3	dump data state
		4	fft state
		5	multiply state
		6	maganitude squared state
		7	generate twiddles state
		8	bit reversal state

Table 8: FFTA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:6 (R/NW)	CTLSTATE0	Thread 0 Control State. The `FFTA_STAT.CTLSTATE0` bit indicates the data available state (reflects state of fft. valid_o).	
		0	idle state
		1	load coef state
		2	load data state
		3	dump data state
		4	fft state
		5	multiply state
		6	maganitude squared state
		7	generate twiddles state
		8	bit reversal state
5 (R/NW)	VLDOUT	Data Available State. The `FFTA_STAT.VLDOUT` bit indicates that data is available in the OFIFO to be read out (reflects state of fft.valid_o).	
		0	Data Not Available
		1	Data Available
4 (R/NW)	RDYOUT	Ready for Data State. The `FFTA_STAT.RDYOUT` bit indicates that the IFIFO is ready to receive data (reflects state of fft.ready_o).	
		0	Data Out Not Ready
		1	Data Out Ready
3 (R/NW)	INVLDINST	An Invalid Instruction Was Executed. The `FFTA_STAT.INVLDINST` bit indicates an invalid instruction was executed. Asserts a FFT_I interrupt output when the INVALIDINST flag is first set. This INVAILIDINST flag is sticky.	
		0	No Invalid Instruction
		1	An Invalid Instruction Was Executed
2 (R/NW)	OVR	Math Result Is Greater Than The Maximum Normalized Number. The `FFTA_STAT.OVR` bit indicates a math result after rounding has a magnitude greater than the maximum normalized number. Asserts a FFT_I interrupt output when the OFLOW flag is first set. This OFLOW flag is sticky.	
		0	No Overflow
		1	Overflow

Table 8: FFTA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	UNDR	Math Result Is Smaller Than The Minimum Normalized Number, But Not A Zero. The `FFTA_STAT.UNDR` bit indicates a math result after rounding has a magnitude less than the minimum normalized number, and it is not an exact zero. Asserts a FFT_I interrupt output when the UFLOW flag is first set. This UFLOW flag is sticky.	
		0	No Underflow
		1	Underflow
0 (R/NW)	INVLDIN	NAN Input Data or Invalid Math Operation. The `FFTA_STAT.INVLDIN` bit indicates a NAN input data or invalid math operation: (0 x infinity, infinity - infinity). Asserts FFT_I interrupt output when the INVALIDIN flag is first set. This INVALIDIN flag is sticky.	
		0	Input And Math Opererations Are Valid
		1	Input And/Or Math Opererations Are Invalid

Thread Count Offset Register

The `FFTA_THREADOFFSET` register is a modulo 3 added to an instructions specified thread number to create the actual thread assignment. The incr_thread_offset() instruction will modulo 3 increment the thread offset register.

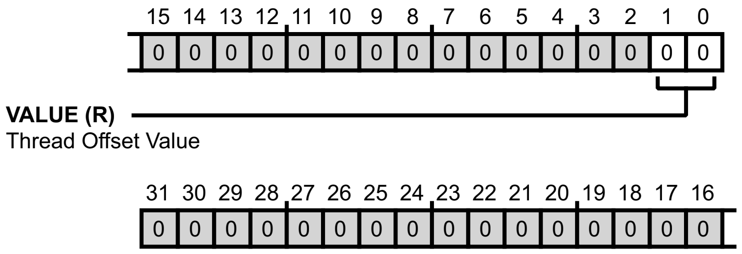

Figure 12: FFTA_THREADOFFSET Register Diagram

Table 9: FFTA_THREADOFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/NW)	VALUE	Thread Offset Value.

Wrapper Control Register

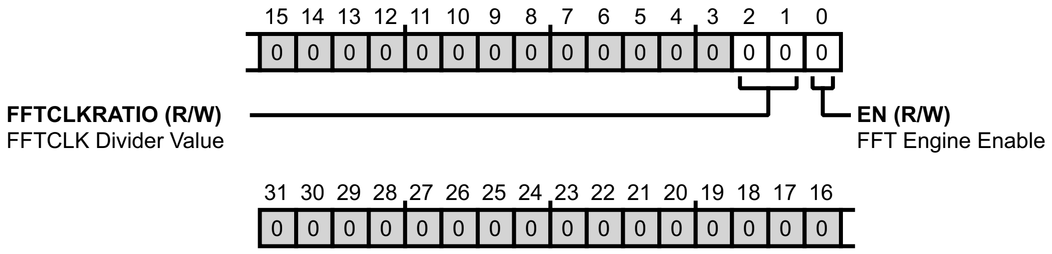

Figure 13: FFTA_WCTL Register Diagram

Table 10: FFTA_WCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:1 (R/W)	FFTCLKRATIO	FFTCLK Divider Value. The `FFTA_WCTL.FFTCLKRATIO` bits provide the FFTCLK divider value which selects the FFTCLK:SYSCLK frequency ratio.	
		0	FFTCLK:SYSCLK = 1:1
		1	FFTCLK:SYSCLK = 1:2
		2	FFTCLK:SYSCLK = 1:4
		3	FFTCLK:SYSCLK = 1:8
0 (R/W)	EN	FFT Engine Enable.	
		0	Disable FFT engine and enable MDMA
		1	Enable FFT engine and Disable MDMA

Load/Dump Transfer Left Register

The three `FFTA_XFRLEFT[nn]` registers, one for each thread, contain the number of 64-bit data or coefficient loads or data dumps remaining if a load or dump instruction is executing in that thread. The value of each register ranges from 0 to 1023, which means (Load/dump transfers remaining for thread)/8. So load/dump remaining can cover the range of [0, 1023*8]=[0, 8184]. If no load or dump is executing in that thread then the value is 0. The values reflect the count at the start of the pipeline (stage N0).

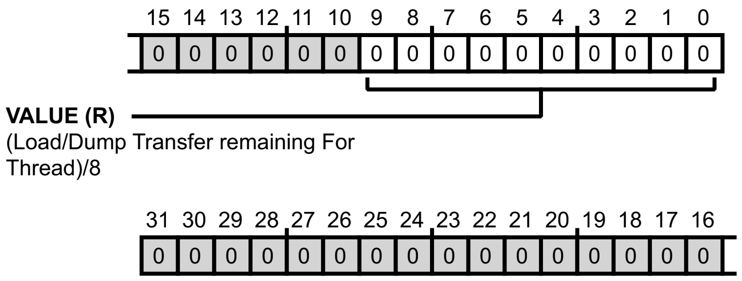

Figure 14: FFTA_XFRLEFT[nn] Register Diagram

Table 11: FFTA_XFRLEFT[nn] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	VALUE	(Load/Dump Transfer remaining For Thread)/8.