50 FIR Accelerator (FIR)

Finite Impulse Response (FIR) filters are frequently used in DSP applications. FIR filters are used in a wide array of applications, and can be used in multi-rate processing in conjunction with an interpolator or decimator. The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core.

Features

This hardware module is capable of performing FIR filters without core intervention. This gives programs freedom to use the core to implement complex algorithms, effectively adding more bandwidth to the processor.

- FIR supports fixed point and IEEE floating-point format
- Has four MAC units which operate in parallel
- Various rounding modes supported
- Single rate or multi-rate window processing
- Change the rates with decimation or interpolation mode
- Up to 32 filter channels available in TDM

NOTE: The FIR accelerator module has local memory which is not accessible by the core during regular operation mode. Unlike previous SHARC processors, the FIR accelerator modules each have access to the system memory (on-chip or off-chip).

Unlike in previous SHARC processors, where only one of the FIR or IIR Accelerator can be enabled at a time, the ADSP-SC58x processor can use both the FIR and the IIR Accelerators at the same time.

Clocking

The FIR accelerator runs at the maximum speed of the system clock frequency (*SCLK0*).

Functional Description

The **FIR Block Diagram** figure shows the block diagram of the 1024-TAP FIR hardware accelerator. The accelerator consists of a 1024 word coefficient memory, a 1024 deep delay line for data, and four MAC units. The accelerator runs at the *SCLK0* frequency.

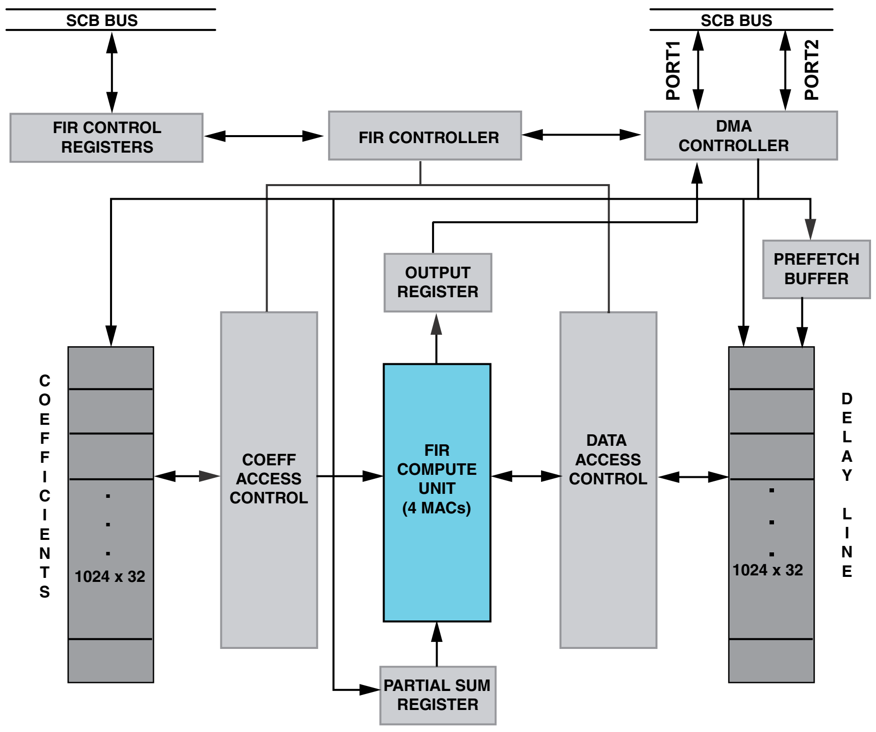

Figure 1: FIR Block Diagram

The FIR accelerator has the following logical sub blocks.

1. A data path unit that consists of:
   - A 1024 deep coefficient memory
   - A 1024 deep delay line for the data
   - Four 32-bit floating-point and fixed-point multiplier and adder units
   - One 32-bit prefetch buffer to operate in a pipelined fashion
   - One 32-bit buffer to hold previous partial sum
   - One 32-bit buffer to hold the output
2. Configuration registers for the number of TAPs, number of channels, filter enable, interrupt control, DMA enable, up sample/down sample control, and ratios.
3. Core access interface for writing the DMA/filter configuration registers and reading the status register.

4. DMA bus interface for transferring data and/or coefficients to and from the accelerator.
5. DMA configuration registers including chain pointer, input, output, and coefficient registers.

ADSP-SC58x FIR Register List

The FIR Accelerator contains the following registers.

Table 1: ADSP-SC58x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register

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

Name	Description
FIR_OUTMOD	FIR Output Data Modifier Register

ADSP-SC58x FIR Trigger List

Table 2: ADSP-SC58x FIR Trigger List Masters

Trigger ID	Name	Description	Sensitivity
60	FIR0_DMA	FIR0DMA	Edge

Table 3: ADSP-SC58x FIR Trigger List Slaves

Trigger ID	Name	Description	Sensitivity
None			

Compute Block

The MAC unit, shown in the following figure, has four multiply accumulators. They operate simultaneously on a single filter as described below.

- The MAC unit operates on the data and coefficient fetched from the data and coefficient RAMs.
- Each MAC can perform 32-bit floating-point or 32-bit fixed-point MAC operations.
- Floating-point format is IEEE compliant.
- Multiply and accumulation operation (addition) are pipelined.
- 32-bit floating-point MAC operation generates 32-bit multiply results.
- 32-bit fixed-point operation generates 80-bit results (64-bit result + 16 guard bits).

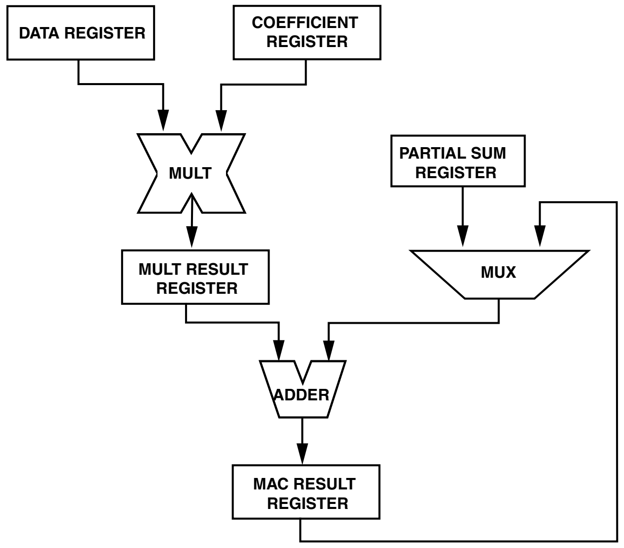

Figure 2: FIR MAC Unit

Partial Sum Register

The partial sum register is useful for floating-point multi-iteration mode. For a particular channel, the intermediate MAC result is written to the system (on-chip or off-chip) memory's output buffer. If the same channel is requested again, the partial result register is updated with the intermediate MAC result via DMA from the system memory's output buffer and added to the current MAC result after each iteration. This process is repeated until all iterations are done (the entire soft filter length is processed).

Delay Line Memory

The accelerator has a 1024 TAP delay line to hold the data locally. The DMA controller fetches the data from system memory and loads it into the delay line. Four read accesses can be made to the delay line simultaneously.

Coefficient Memory

The accelerator has a 1024 deep coefficient memory to store the coefficients. The DMA controller loads the coefficients from system memory into coefficient memory. Four coefficients can be fetched from the coefficient memory simultaneously. If the soft filter length is more than 1024, processing is done in multi-iteration mode.

Pre Fetch Data Buffer

This buffer enables pipeline operation. One data sample is pre-fetch when the compute unit is operating on the delay-line corresponding to the current sample. The data pre-fetched in this buffer is later used to update the delay line for the next sample. This happens in parallel again, when the compute unit is not accessing the delay line in other words when it is adding the output from the four MACs and the partial sum register.

Table 4: Pipeline Operation for Window Size = 1

Cycles	1	2	3	4	5	6
Output DMA			N	N1	N2	N3
Compute		N	N1	N2	N3	
Input DMA	N	prefetch N1	prefetch N2	prefetch N3		

Processing Output

The accelerator uses all four MACs simultaneously to calculate one output sample as shown in the **Multi-Iteration Filtering Flow** figure and the following procedure.

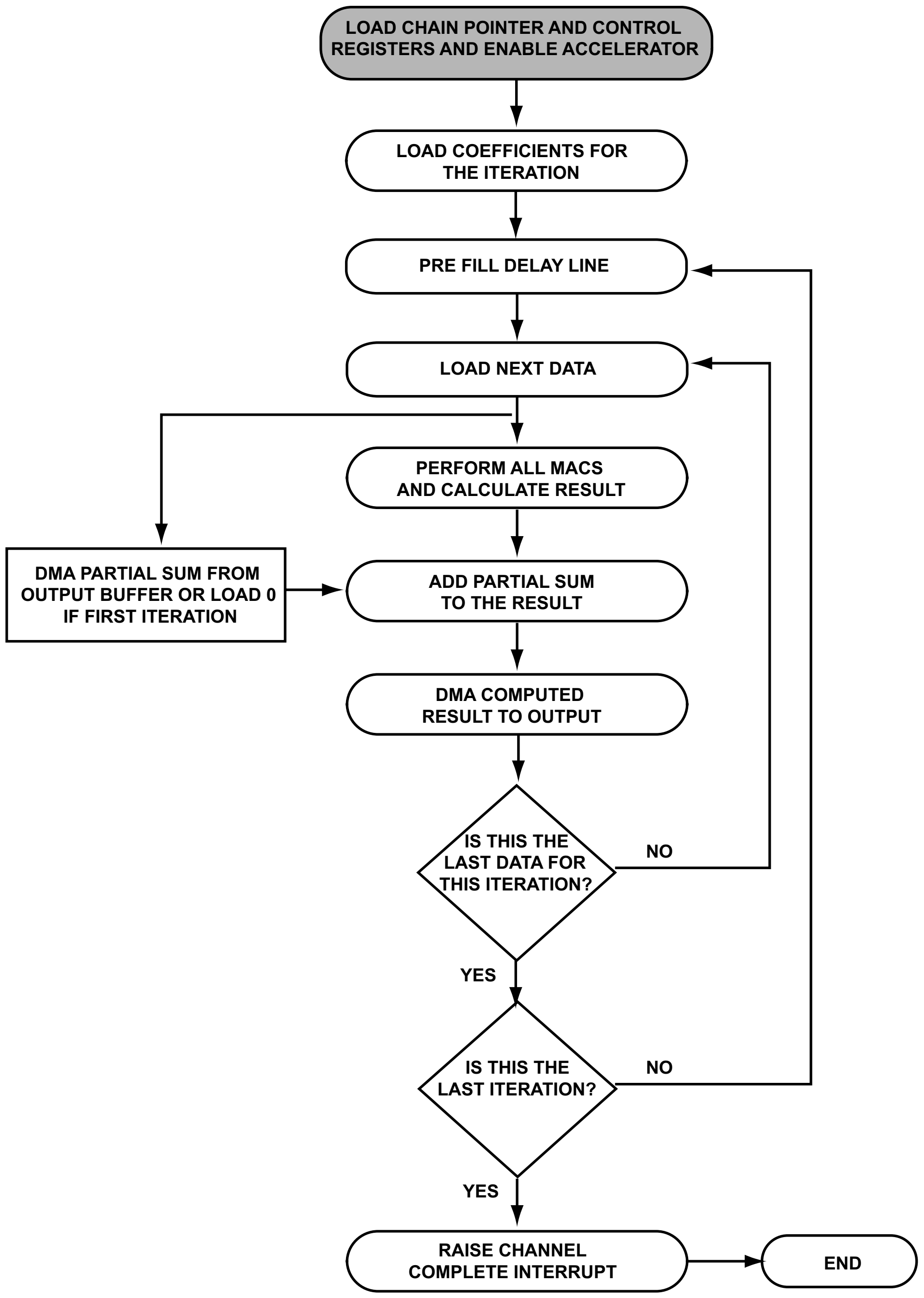
LOAD CHAIN POINTER AND CONTROL REGISTERS AND ENABLE ACCELERATOR
LOAD COEFFICIENTS FOR THE ITERATION
PRE FILL DELAY LINE
LOAD NEXT DATA
PERFORM ALL MACS AND CALCULATE RESULT
DMA PARTIAL SUM FROM OUTPUT BUFFER OR LOAD 0 IF FIRST ITERATION
ADD PARTIAL SUM TO THE RESULT
DMA COMPUTED RESULT TO OUTPUT
IS THIS THE LAST DATA FOR THIS ITERATION?
NO
YES
IS THIS THE LAST ITERATION?
NO
YES
RAISE CHANNEL COMPLETE INTERRUPT
END

Figure 3: Multi-Iteration Filtering Flow

1. The accelerator fetches four input data from the delay line and four corresponding coefficients from the coefficient memory and feeds them to the MAC units for multiply/accumulation.
2. The accelerator repeats the procedure with the next four input data and coefficients until all the TAPs complete. For an N TAP filter for example, this procedure is done N/4 times.

3. When all the TAPs complete, the accelerator adds the four MAC outputs together to the previous partial sum (if any) to calculate the final result.
4. Finally, that output sample is stored back in system memory.

System Memory Storage

The following sections describe the storage format for the accelerator.

CAUTION: Store all data in memory aligned to the word address boundaries. Any other programmed addresses do not flag an error.

Coefficients and Input Buffer Storage

For any N TAP filter with coefficients:

```
C[i] i = 0,1,
...
N - 1
```

the coefficients should be stored in system memory buffer in the order:

```
C[N - 1], C[N - 2]
...
C[1],C[0]
```

and the CI should point to `C(N - 1)`

Single Rate Input Filtering

The total size of the input buffer should at least be equal to N - 1 + W. If the input buffer that needs to be processed is:

```
x[n],x[n+1],x[n+2]
...
x[n+W-1]
```

it should be stored in the memory as

```
x[n-(N-1)], x[n-(N-2)]
... x[n-1], x[n], x[n+1]
...
x[n+W-1]
```

and FIR_INIDX should point to `x[n - (N - 1)]`

Decimation

Assuming M = decimation ratio, the total size of the input buffer should at least be equal to `N-1+WxM`. If the input buffer that needs to be processed is

```
x[n],x[n+1],x[n+2]....x[n+WxM-1],
```

it should be stored in the memory as
```
x[n-(N-1)], x[n-(N-2)]....x[n-1],
x[n], x[n+1]....x[n+WxM-1]
```

and FIR_INIDX should point to `x[n-(N-1)]`.

Interpolation

Assuming L= interpolation ratio, the total size of the input buffer should at least be equal to Ceil `((N-1)/L)+W/L`.

If the input buffer that needs to be processed is
```
x[n], x[n+1], x[n+2]....x[n+W/L-1],
and K= Ceil((N-1)/L)
```

it should be stored in the memory as:
```
x[n-k], x[n-(K-1)], x[n-(K-2)]....x[n-1],
x[n], x[n+1].... x[n+W/L-1]
```

and the FIR_INIDX should point to `x[n-K]`.

Operating Modes

The FIR core performs a sum-of-products operation to compute the convolution sum. It supports single-rate, decimation, and interpolation functions.

Single Rate Processing

In a single-rate filter, the output result rate is equal to the input sample rate. The filter output Y(n) is computed according to following equation where N is the number of filter coefficients: c(i) i = 0,..., N - 1 are the filter coefficients and x(n) represents the input time-series.

$$Y(n)= \sum_{k=0}^{N-1} c(k) \times x(n-k)$$

Figure 4: Filter Output Calculation

Single Iteration

Results are computed in single iteration when the soft filter length is less than or equal to 1024.

Multi Iteration

Results are computed in multiple iterations when the soft filter length is greater than 1024 (for example, 2048 TAPs on a 1024 hard filter length). In this mode, the controller implements two iterations of 1024

TAPs. Note that if the soft filter length is not a multiple of the hard filter length the controller iterates until the soft filter length is satisfied.

Example: 550 taps on a 256 tap filter.

In this example, the FIR controller implements two iterations of 256 taps and one iteration of 38 taps.

NOTE: Multi-iteration mode is not supported in fixed-point format.

Window Processing

Sample based processing mode is selected by configuring window size to 1. In this mode, one sample from a particular channel is processed through all the biquads of that channel and the final output sample is calculated.

In window based mode, multiple output samples (up to 1024) equal to the window size of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A configurable window size parameter is provided to specify the length of the window.

Multi Rate Processing

Multi rate filters change the sampling rate of a signal-they convert the input samples of a signal to a different set of data that represents the same signal sampled at a different rate.

Decimation

A decimation filter provides a single output result for every M input samples, where M is the decimation ratio. Note that the output rate is 1/M'th of the input rate. The filter implementation exploits the low output sample rate by not starting a computation until a new set of M input samples is available.

In this mode, after low pass filtering (for anti aliasing), FIR logic discards the ratio - 1 samples of output data. For performance optimization, FIR logic skips the computation of output samples, which are discarded.

The input buffer size for decimation filters is N - 1 + (W × M) where:

- N is the number of taps
- W is the window size
- M is the decimation ratio

The window size (`FIR_CTL2.WINDOW` bits) must be programmed with the number of output samples.

To start this mode, programs set the FIR_CTL2.RATIO and FIR_CTL2.UPSAMP bits (along with normal filter setting). Also the FIR_CTL2.TAPLEN bit field value should be greater than or equal to the FIR_CTL2.RATIO bit field value for decimation filter.

Interpolation

An interpolation filter provides L output results for each new input sample, where L is the interpolation ratio. Note that the output rate is L times the input rate.

In this mode, according to the ratio specified in configuration register, FIR logic inserts L – 1 zeros between any two input samples (up-sampling) and then performs the interpolation (through the FIR filter).

Both up-sampling and down-sampling do not support multi iteration mode. Therefore, the filtering operation can only be done on up to 1024 TAPs and the ratio of up/down sampling can only be an integer value.

In an interpolation filter FIR logic inserts L - 1 zeros between each sample and the program has to make sure that these zeros are fully shifted out of the delay line before moving on to the next channel. This puts a restriction on window size in terms of L - *the sample ratio* as shown below.

$WINDOWSIZE = n \times SAMPLERATIO$ where n is the number of input samples.

The input buffer size is smallest integer greater than or equal to (N – 1 + W)/L for interpolation filters where:

- N is the number of taps
- W is the window size
- L is the interpolation ratio

To start the mode, programs configure the FIR_CTL2.RATIO and FIR_CTL2.UPSAMP bits (along with filter settings).

Channel Processing

The **Single Channel Filtering Flow** figure shows the flow diagram for processing a single channel. Channels are processed in TDM format by setting the FIR_CTL1.CH bits to a value greater than one. In the time slot corresponding to a particular channel, the corresponding TCB is loaded from system memory.

1. The FIR_CTL2 register value is fetched from system memory and is used to configure the filter parameters for that channel.
2. The accelerator fetches the coefficients using the FIR_COEFIDX register as the pointer and loads them into coefficient memory.
3. The delay line is pre-filled using the FIR_INIDX register as the pointer.

4. The accelerator calculates the first output and stores the result back into the output buffer using the `FIR_OUTIDX` register as the pointer.
5. While calculating the output the accelerator fetches the next data in parallel. After one window of data is processed, the index registers in the system memory TCB ares updated so that in the next time slot of the same channel, processing can be continued from where it stopped.
6. Processing moves to the next channel and repeats the procedure. If the soft filter length is more than the hard filter length, multiple iterations are done to process the window.

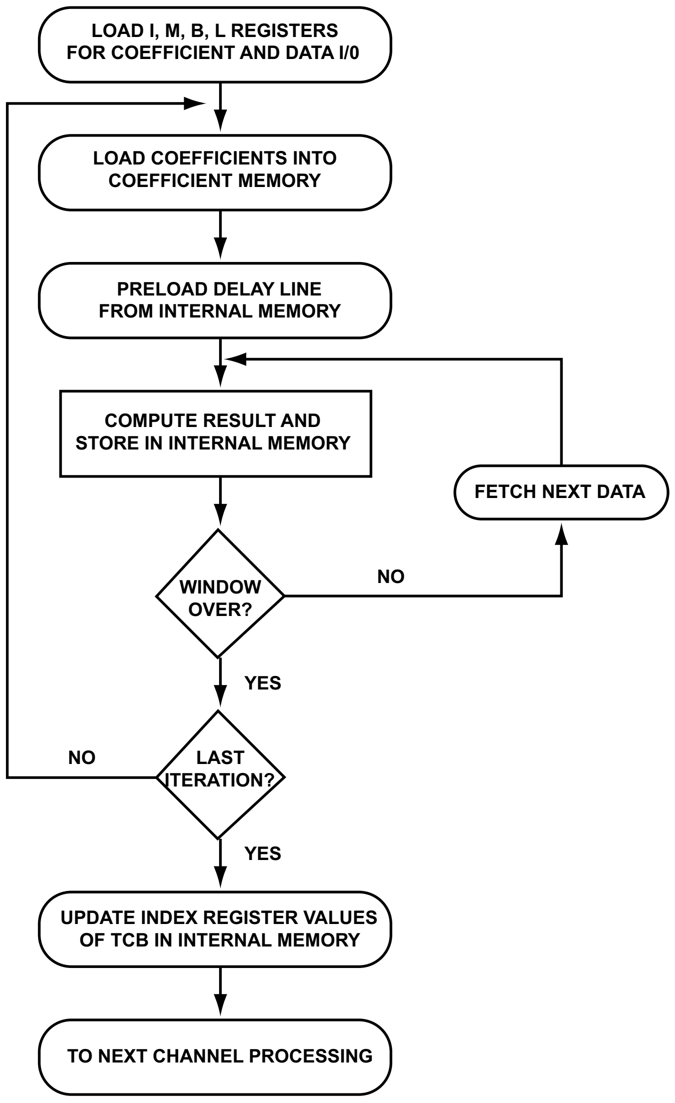
LOAD I, M, B, L REGISTERS FOR COEFFICIENT AND DATA I/0
LOAD COEFFICIENTS INTO COEFFICIENT MEMORY
PRELOAD DELAY LINE FROM INTERNAL MEMORY
COMPUTE RESULT AND STORE IN INTERNAL MEMORY
FETCH NEXT DATA
WINDOW OVER?
NO
YES
LAST ITERATION?
NO
YES
UPDATE INDEX REGISTER VALUES OF TCB IN INTERNAL MEMORY
TO NEXT CHANNEL PROCESSING

Figure 5: Single Channel Filtering Flow

Floating-Point Data Format

The FIR accelerator treats data and coefficients in 32-bit floating-point format as the default functional mode.

Fixed-Point Data Format

In fixed-point mode, the 32-bit input data/coefficient is treated as fixed-point. A 32-bit fixed-point MAC operation generates an 80-bit result. Fixed-point data/coefficients can be unsigned integer, unsigned fractional and signed integer.

NOTE: In fixed point mode, the entire 80-bit result register is always written back in bursts of 3 × 32 bits. The first word is the LSW, the 2nd the MSW and the third word is a 16-bit overflow, the remaining 16-bits are padded with zeros. Therefore for fixed-point WINDOWSIZE = WINDOWSIZE × 3.

If signed fractional format is used, the output needs to be scaled by 2 since the MAC does not the right shift to remove the redundant sign bit. A final routine needs to decimate the output buffer to the desired samples.

Multi iteration mode is not supported in this format. Therefore, the maximum TAP length is 1024.

Data Transfer

The FIR filter works exclusively through DMA.

Chain Assignment

The structure of a TCB is conceptually the same as that of a traditional linked-list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB may point to itself to continuously re-run the same DMA. The FIR accelerator reads each word of the TCB and loads it into the corresponding register. The end of the chain (no further TCBs are loaded) is indicated by a TCB with a chain pointer register value of zero.

The FIR accelerator DMA supports circular buffer chained DMA. The FIR accelerator does not support circular buffering for the coefficient buffer. The **TCBs for Chained DMA** table shows the required TCBs for chained DMA.

Table 5: TCBs for Chained DMA

Address	Register
TCB	FIR_CHNPTR
TCB + 0x1	FIR_COEFCNT
TCB + 0x2	FIR_COEFMOD

Table 5: TCBs for Chained DMA (Continued)

Address	Register
TCB + 0x3	FIR_COEFIDX
TCB + 0x4	FIR_OUTBASE
TCB + 0x5	FIR_OUTCNT
TCB + 0x6	FIR_OUTMOD
TCB + 0x7	FIR_OUTIDX
TCB + 0x8	FIR_INBASE
TCB + 0x9	FIR_INCNT
TCB + 0xA	FIR_INMOD
TCB + 0xB	FIR_INIDX
TCB + 0xC	FIR_CTL2

The FIR_COEFCNT register is loaded with the values in the FIR_COEFCNT TCB field and is decremented from that value onwards. However, coefficient loading continues until the number of coefficients, equal to the tap length, are read. This is true even if the FIR_COEFCNT register reaches zero as in the case of a tap length = 10, and theFIR_COEFCNT field in the TCB is initialized to 0. The value in the FIR_COEFCNT register is –10 after all coefficients are loaded.

NOTE: Initialize FIR_CHNPTR to TCB+12.

DMA Access

The FIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register that points to the next set of DMA parameters stored in the processor's internal memory. In chained DMA operations, the processor automatically initializes and then begins another DMA transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Pointer DMA

The DMA controller supports circular buffer chain pointer DMA. One transfer control block (TCB) needs to be configured for each channel. The TCB contains:

- A control register value to configure the filter parameters (such as filter tap length, window size, sample rate conversion settings) for each channel

- DMA parameter register values for the input data (delay line)
- DMA parameter register values for coefficient load
- DMA parameter register values for output data

Intermediate results in multi-iteration mode are saved in the output buffer

As shown in the **Circular Buffer Addressing** figure, the accelerator loads the TCB into its internal registers and uses these values to fetch coefficients and data and to store results. After processing a window of data for any channel, the accelerator writes back the appropriate values to the `FIR_INIDX` and `FIR_OUTIDX` fields of the TCB in memory, so that data processing can begin from where it left off during the next time slot of that channel.

The write back value for input buffer is:

- `FIR_INIDX` + W for single rate filtering.
- `FIR_INIDX` + W × M for decimation (M = decimation ratio).
- `FIR_INIDX` + W/L for interpolation (L = interpolation ratio).
- The write back values for output buffer in floating point mode is: `FIR_OUTIDX` + W.
- The write back values for output buffer in fixed point mode is: `FIR_OUTIDX` + 3 × W.

NOTE: The `FIR_CTL2` register is part of the FIR TCB. This allows programming individual FIR channels with different control attributes.

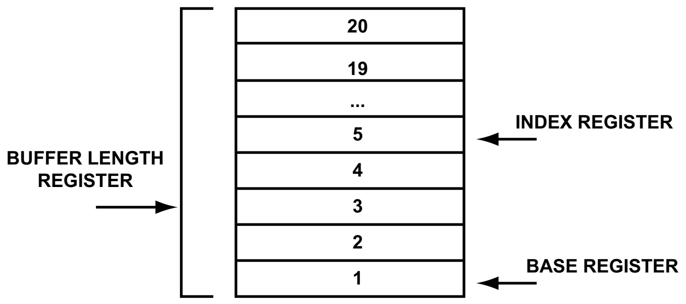

Figure 6: Circular Buffer Addressing

Programming Model

The following sections provide general programming information for the FIR accelerator.

Single Channel Processing

1. Create input, coefficient, and output buffers in system memory.

   For input and coefficient buffer storage format see the “Coefficients and Input Buffer Storage” section.

2. Create the TCBs in system memory. Each TCB corresponds to a particular channel.

   TCBs hold the `FIR_CTL2` register which allows programming the window size and tap size along with up or down sample enable, sample rate conversion enable, and the conversion ratio for decimation and interpolation filters.

3. Configure the index, modifier, length entries in the TCBs to point to the corresponding channels' data buffer, coefficient buffer, and output data buffer.

   The output index register should always point to the start of the output buffer. However, the input index register's value should be initialized based on the explanation provided in the “Coefficients and Input Buffer Storage” section.

4. The core configures the `FIR_CTL1` register with the number of channels (one channel), fixed- or floating-point format.

5. Set the enable bit to start accelerator operation in the modes configured (in `FIR_CTL1` and `FIR_CTL2` registers) by loading the first channels’ TCB. Once the first channel window is calculated, the input and output index registers are written back to internal memory corresponding to the first channel. Once the write back is complete the accelerator moves into idle.

NOTE: All the addresses programmed in the TCB should correspond to 32-bit address boundaries and should not contain the lower 2 bits (assumed as zeros).

Multichannel Processing

The **Wait for Core Intervention => Idle (if CAI bit = 0)** figure shows the diagram for multichannel filtering. Multiple channels are processed in a time division multiplexed (TDM) format. After completing all the channels, the accelerator can either repeat the slots or wait for core intervention.

For multichannel filtering, use the following steps.

1. Program the number of channels using the `FIR_CTL1.CH` bits.

2. Configure the TCBs in system memory with one channel's TCB pointing to the next channel's TCB.

3. Write the first TCB value into the `FIR_CHNPTR` register and enable the accelerator.

   The accelerator fetches the first channel's TCB and, using it as pointer, pre-fills the delay line and coefficient memory and loads the `FIR_CTL2` register to configure the filter parameters corresponding to that channel.

   The accelerator then calculates output samples corresponding to one Window and stores the data back in internal memory.

   At the end of the Window the accelerator updates the`FIR_INIDX` and `FIR_OUTIDX` registers in the TCB of system memory and moves to the next channel.

   When all the channels are finished and the auto channel iterate bit (`FIR_CTL1.CAI`) is set, the accelerator processes the first channel again and iterates through the channels. If the `FIR_CTL1.CAI` bit is cleared, the accelerator waits for core intervention.

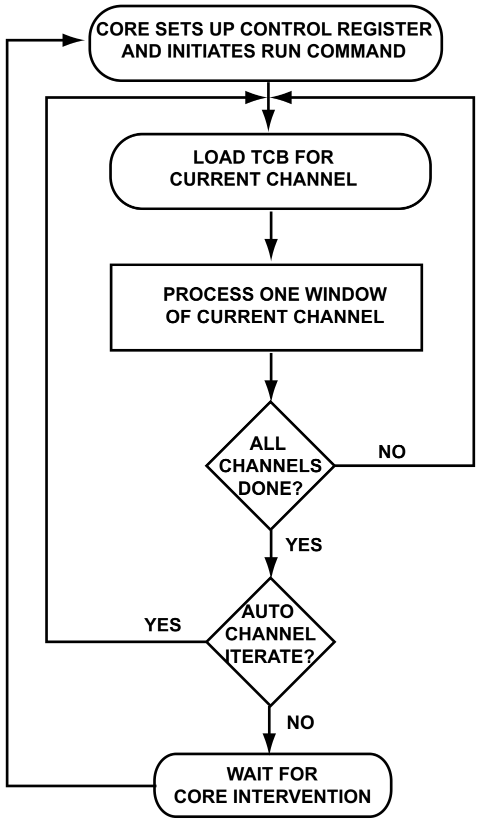
CORE SETS UP CONTROL REGISTER AND INITIATES RUN COMMAND
LOAD TCB FOR CURRENT CHANNEL
PROCESS ONE WINDOW OF CURRENT CHANNEL
ALL CHANNELS DONE?
NO
YES
AUTO CHANNEL ITERATE?
YES
NO
WAIT FOR CORE INTERVENTION

Figure 7: Wait for Core Intervention => Idle (if CAI bit = 0)

Dynamic Coefficient Processing Notes

1. The dynamic update of the coefficients may be useful for the FIR accelerator. The reason is that the FIR accelerator re-loads the coefficients for each iteration (if the `FIR_CTL1.CAI` bit is set) before the start of processing of each channel.
2. The dynamic coefficient update should be possible for single iteration mode (tap length ≤1024) by making sure that the new coefficients are updated after the accelerator loads the coefficients for current processing and before the next processing starts. The expression for the maximum time available for the coefficient memory update should be equal to TBD.
3. For multi-iteration mode dynamic updates are not supported. Programs must finish current processing, disable the accelerator, update the coefficients, and re-enable the accelerator.

Debug Mode

The next sections show the steps required for reading and writing local memory in debug mode.

Write to Local Memory

1. Clear the `FIR_CTL1.DMAEN` bit.
2. Set the `FIR_DBG_CTL.EN`, `FIR_DBG_CTL.MEM` and `FIR_DBG_CTL.HLD` bits.
3. Set the `FIR_DBG_CTL.ADRINC` bit for address auto increment.
4. Write the start address to the `FIR_DBG_ADDR` register. Note if bit 11 is set, coefficient memory is selected.
5. Wait at least TBD cycles.
6. Write data to the `FIR_DBG_WRDAT` register.

Read from Local Memory

1. Clear the `FIR_CTL1.DMAEN` bit.
2. Set the `FIR_DBG_CTL.EN`, `FIR_DBG_CTL.MEM` and `FIR_DBG_CTL.HLD` bits.
3. Set the `FIR_DBG_CTL.ADRINC` bit for address auto increment.
4. Write the start address to the `FIR_DBG_ADDR` register. Note if bit 11 is set, coefficient memory is selected.
5. Wait at least TBD cycles.
6. Read data from the `FIR_DBG_RDDAT` register.

Single Step Mode

Single step mode can be used for debug purposes. An additional debug register is used in this mode.

1. Enable stop DMA during breakpoint hit in the emulator settings.
2. Clear the `FIR_DBG_CTL.HLD` bit and enable `FIR_DBG_CTL.EN` and `FIR_DBG_CTL.RUN` bits.
3. Program FIR module according to the application.
4. In single step each iteration is updated in the emulator session.

FIR Programming Example

An application needs FIR filtering of six channels of data. The first four channels require 256 TAP filtering and the last two channels require 1024 TAP filtering. The window size for all the channels is 128.

1. Create a circular data buffer in system memory for each channel.

   The buffer should be large enough to avoid overwriting data before being processed by the accelerator. Ideally, the input buffer size for a channel is*tap length* + *window size* - 1 for that channel. The 256 coefficients of each of the first four channels and the 1024 coefficients each of the last two channels are also configured in system memory buffers. The output buffer size is equal to the window size.
2. Create six TCBs in system memory with each channel's chain pointer (CP) entry pointing to the next channel's and the sixth channel's CP entry pointing back to the first channel's in a circular fashion.
3. Configure the `FIR_CTL2` register for the first four channels' TCBs to 256 TAPs and a window size of 128, and the next two channels for 1024 TAPs and a window size of 128, respectively.
4. Configure the index, modifier, length entries in the TCBs to point to the corresponding channel's data buffer, coefficient buffer, and output data buffer. The location of the first channel's TCB is written to the `FIR_CHNPTR` register. The `FIR_CTL1.CH` bit field is then programmed with an value that corresponds to six channels.
   a. a. The accelerator iterates through six channels once and then waits for core intervention, (the `FIR_CTL1.CAI` bit is not set, the DMA is enabled, and the `FIR_CTL1.EN` bit is set).
   b. The accelerator then loads the first channel's TCB then loads the coefficient and data and processes one window.
   c. After saving the index values to memory the accelerator moves to the next channel.
   d. After all six channels are complete the accelerator halts and waits for core intervention.

Computing FIR Output, Tap Length > Than 4096

With little core intervention, the FIR accelerator can as well be used to calculate output for tap length greater than 4096 taps. The section shows how it can be done with an example of 8192 taps. Transfer function of an 8192 FIR filter can be divided into two 4096 FIR filters as shown below.

$H(Z) = b_0 + b_1Z^{-1} + b_2Z^{-2} + \ldots\ldots\ldots\ldots b_{4095}Z^{-4095} + b_{4096}Z^{-4096}\mathrm{b}_{4097}Z^{-4097} + \ldots\ldots\ldots b_{8191}Z^{-8191}$

$= b_0 + b_1Z^{-1} + b_2Z^{-2}+ \ldots\ldots\ldots\ldots b_{4095}Z^{-4095} + Z^{-4096}(b_{4096}+b_{4097}+ \ldots\ldots\ldots b_{8191}Z^{-4096})$

Filter coefficients of an 8192 tap filter therefore need to be divided among two 4096 tap FIR filters.

Filter 1

Coefficients = b0, b1, b2,……, b4095

Input data = x[n], x [n-1],……….x[n-4095]

Filter 2

Coefficients=b4096, b4097,…………..,b8191

Input data = x[n-4096], x[n-4097],……x[n-8191]

The accelerator can be used in two channel mode where channel 1 operates on x[n]…x[n-4095] input data with the filter coefficients of filter 1 and channel 2 operates on x[n-4096]…x[n-8191] with the filter coefficients of filter 2.

Once both the channels are processed, the partial sum output of both the channels can to be added together to get the final output. The following programming steps are needed to implement this approach (tap length = TAPS = 8192, window size = WINDOW).

1. Create a circular input data buffer in system memory (IBUF). The buffer should be large enough to avoid overwriting data before being processed by the accelerator. Ideally, the input buffer size for a channel is TAPS + WINDOW – 1.
2. Create a coefficient buffer of size TAPS (8192) (CBUF).
3. Create one output buffer of size WINDOW (OBUF) and another temporary output buffer (OBUF1) to store the partial sum.
4. Create two TCBs in system memory with first TCB chained to the second and second one chained to the first in circular manner.
   a. The `FIR_COEFIDX` field of the first TCB should point to the start address of the coefficient buffer (CBUF) and that of the second TCB should point to 4096 offset from the start of the coefficient buffer (CBUF + 4096).
   b. The `FIR_OUTBASE` and `FIR_OUTIDX` field of the first TCB should point to the start address of OBUF and that of the second TCB should point to the start address of OBUF1.

   c. The FIR_INIDX field of the first TCB should point to the start address of IBUF and that of the second TCB should point to 4096 offset from the start address of IBUF.

   d. The FIR_CTL2 field of both the TCB should be configured for tap length = TAP/2 = 4096 and window size = WINDOW.

5. Initialize the FIR_CHNPTR register pointing to the first TCB.
6. Program the FIR_CTL1 register to initiate the accelerator processing now by setting the FIR_CTL1.EN and FIR_CTL1.DMAEN bits and the number of channels configured as 2.
7. Wait for the FIR all channel done interrupt to occur and inside the ISR, add the partial sum results using core from both the output buffers (OBUF and OBUF1) to get the final output. To save memory, the contents of the buffer OBUF can be replaced by the final output result.

Debug Features

The following sections provide information of debugging the FIR accelerator.

Local Memory Access

The contents of FIR delay line and coefficient memories are made observable for debug by setting the FIR_DBG_CTL.EN/FIR_DBG_CTL.MEM and FIR_DBG_CTL.HLD bits. The debug address register (FIR_DBG_ADDR) and two data registers are provided for debug operations. Bit 11 of the FIR_DBG_ADDR register selects coefficient memory if set (=1) and selects delay line memory in cleared (=0).

In the debug mode, the read data register (FIR_DBG_RDDAT) returns the contents of the memory location pointed to by the address register. Data can be written into any memory location using FIR_DBG_WRDAT register writes. If the address auto increment bit (FIR_DBG_CTL.ADRINC) is set, the address register auto increments on FIR_DBG_WRDAT writes and FIR_DBG_RDDAT reads. During auto increment, the FIR_DBG_ADDR register cannot cross the data memory/ coefficient memory boundary.

Single Step Mode

Programs can single step through the MAC operations and observe the memory contents after each step. The FIR_DBG_CTL.EN, FIR_DBG_CTL.HLD, and FIR_DBG_CTL.MEM bits control the FIR MAC units.

Emulation Considerations

In FIR debug mode, the DMA operations are not observable.

Interrupts

The following provides an overview of FIR interrupts.

Table 6: FIR Interrupt Overview

Default Programmable Interrupt	Sources	Masking	Service
FIR_DMA FIR_STAT	Input DMA complete Output DMA complete Window complete All channels complete	N/A	ROC from FIR_DMASTAT + RTI instruction
	MAC IEEE floating-point exceptions MAC fixed-point Overflow		ROC from FIR_MACSTAT + RTI instruction

Sources

The FIR module drives two interrupt signals, `FIR_DMA` for the DMA status and `FIR_STAT` for the MAC status. The FIR module generates interrupts as described in the following sections.

Window Complete

This interrupt is generated at the end of each channel when all the output samples are calculated corresponding to a window and updated index values are written back.

All Channels Complete

This interrupt is generated when all the channels are complete or when one iteration of time slots completes. Note that the interrupt follows the access completion rule, where the interrupt is generated when all data are written back to system memory.

MAC Status

A MAC status interrupt is generated under the following conditions and is reflected in the `FIR_MACSTAT` register.

- Multiplier result zero – Set if Multiplier result is zero
- Multiplier Result Infinity – Set if Multiplier result is Infinity
- Multiply Invalid – Set if Multiply operation is Invalid
- Adder result zero – Set if Adder result is zero

- Adder result infinity – Set if Adder result is infinity
- Adder invalid – Set if Addition is invalid
- Adder overflow – for fixed-point operation

Service

When a DMA interrupt occurs, programs can find whether the input or output DMA interrupt occurred by reading the DMA status register (`FIR_DMASTAT`). The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this by reading the MAC status register (`FIR_MACSTAT`). The MAC interrupt status bits are sticky and are cleared by a read.

The status interrupt sources are derived from the `FIR_MACSTAT` register. If the status interrupt occurs as a result of the last set of MAC operations of a processing iteration corresponding to a particular channel, the interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped by another processing iteration that results in a non-zero or valid multiply/add result. However, in this situation it is difficult to isolate whether the interrupt corresponds to the previous processing iteration or that of the current one. This makes the use of status interrupts impractical.

An alternate way is to poll status bits of the `FIR_MACSTAT` register inside the DMA interrupt service routine. However, the behavior of the status bits, as described below, should be kept in mind. The status bits in the `FIR_MACSTAT` registers are sticky. Once a status bit is set, it gets cleared only when the `FIR_MACSTAT` register is read and the previous set of MAC operations resulted in a non-zero/valid output. Therefore, if the last set of MAC operations of a particular processing iteration results in a zero/non-valid output, the corresponding status bit aren't cleared, even after reading the `FIR_MACSTAT` register. To avoid a false indication in the next processing iteration, it is necessary to ensure that all the status bits are cleared after the current iteration finishes.

The solution is to read the `FIR_MACSTAT` register twice inside the DMA interrupt service routine. The first read is used to identify which status bits are set. The second read is used to discover if the status bit was set because of the last set of MAC operations. If the status bit was not set because of the last set of MAC operations, it provides a zero result.

Otherwise, the bit was set because of the last set of MAC operations. In that case, the status bit must be cleared by performing a simple dummy FIR processing iteration (tap length = 4 and window size = 1) by choosing the appropriate coefficients and input buffer and reading the `FIR_MACSTAT` register after the processing is complete.

For more information see the "FIR MAC Status Register" section.

Effect Latency

The total effect latency is a combination of the write effect latency (core access) plus the peripheral effect latency (peripheral specific).

Write Effect Latency

For details on write effect latency, see *SHARC Processor Programming Reference.*

FIR Accelerator Effect Latency

TBD

ADSP-SC58x FIR Register Descriptions

The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core. (FIR) contains the following registers.

Table 7: ADSP-SC58x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register

Table 7: ADSP-SC58x FIR Register List (Continued)

Name	Description
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register
FIR_OUTMOD	FIR Output Data Modifier Register

FIR Chain Pointer Register

The FIR_CHNPTR register contains the chain pointer address.

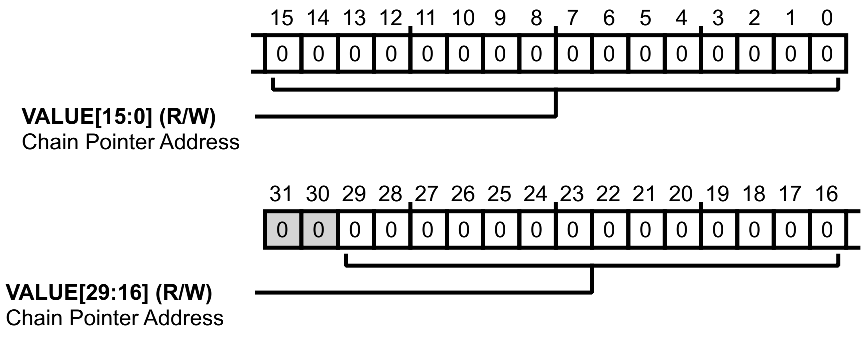

Figure 8: FIR_CHNPTR Register Diagram

Table 8: FIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Chain Pointer Address. The FIR_CHNPTR.VALUE bit field contains the chain pointer address.

FIR Coefficient Count Register

The FIR_COEFCNT register contains the 16-bit coefficient buffer count.

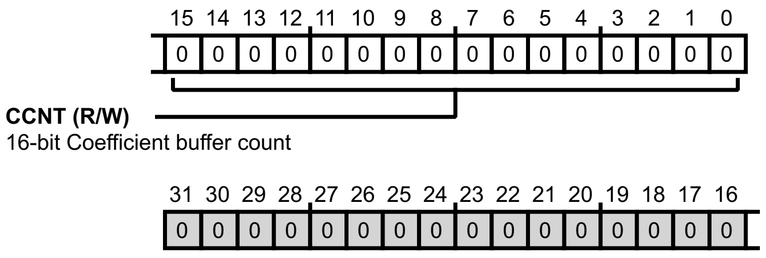

Figure 9: FIR_COEFCNT Register Diagram

Table 9: FIR_COEFCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CCNT	16-bit Coefficient buffer count. The FIR_COEFCNT.CCNT bit field contains the 16-bit Coefficient buffer count.

FIR Coefficient Index Register

The FIR_COEFIDX register contains the coefficient index word address with the lower two bits removed.

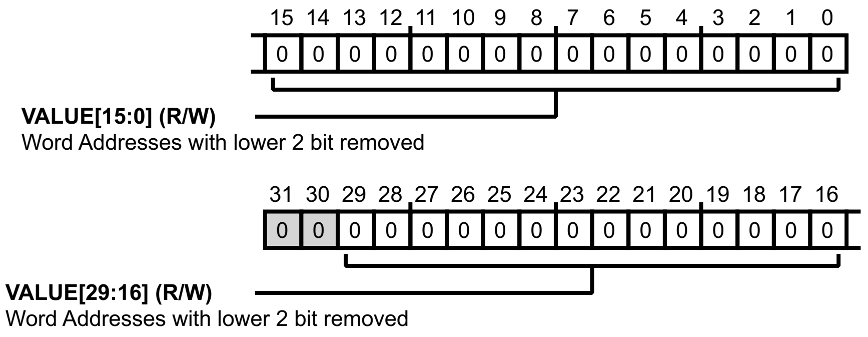

Figure 10: FIR_COEFIDX Register Diagram

Table 10: FIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Addresses with lower 2 bit removed. The FIR_COEFIDX.VALUE bit field contains the word addresses with the lower 2 bits removed.

FIR Coefficient Modifier Register

The `FIR_COEFMOD` register contains the 16-bit coefficient index modifier.

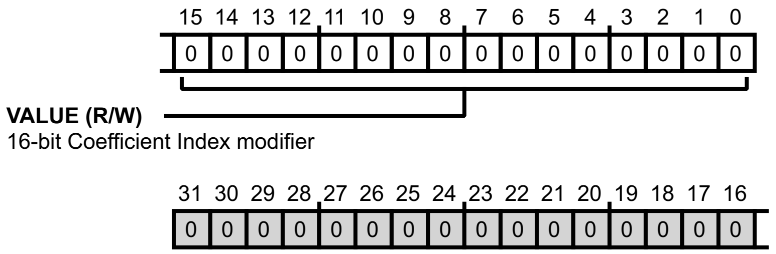

Figure 11: FIR_COEFMOD Register Diagram

Table 11: FIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Coefficient Index modifier. The `FIR_COEFMOD.VALUE` bit field contains the 16-bit Coefficient Index modifier.

FIR Global Control Register

The `FIR_CTL1` register is used to configure the global parameters for the accelerator. These include the number of channels, channel auto iterate, DMA enable, and accelerator enable.

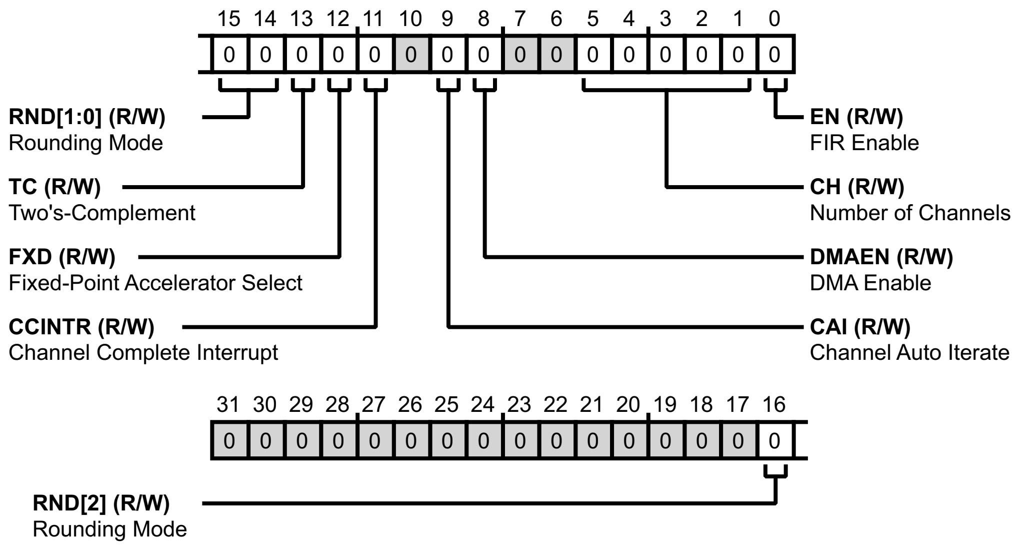

Figure 12: FIR_CTL1 Register Diagram

Table 12: FIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16:14 (R/W)	RND	Rounding Mode. The `FIR_CTL1.RND` bit configures the accelerator to use one of the following rounding modes.	
		0	IEEE round to nearest (even)
		1	IEEE round to +ve infinity
		2	IEEE round to -ve infinity
		3	Round to nearest Up
		4	Round away from zero
		5	Reserved
		6	Reserved
13 (R/W)	TC	Two's-Complement. The `FIR_CTL1.TC` bit configures the accelerator to use either unsigned integer or signed integer	
		0	Unsigned integer
		1	Signed integer
12 (R/W)	FXD	Fixed-Point Accelerator Select. The `FIR_CTL1.FXD` bit configures the accelerator to use either 32-bit IEEE floating-point or 32-bit fixed-point.	
		0	32-bit IEEE floating-point
		1	32-bit fixed point
11 (R/W)	CCINTR	Channel Complete Interrupt. The `FIR_CTL1.CCINTR` bit configures the accelerator to generate an interrupt when each or all channels are done.	
		0	Interrupt is generated only when all channels are done
		1	Interrupt is generated after each channel is done
9 (R/W)	CAI	Channel Auto Iterate. The `FIR_CTL1.CAI` bit if cleared causes TDM processing to stop (idle) once all channels are over. If set, processing moves to the first channel and continues TDM processing in a loop when all channels are over.	
		0	TDM Processing stops (idle) once all channels are over
		1	Moves to first channel and continues TDM processing in a loop when all channels are over
8 (R/W)	DMAEN	DMA Enable. The `FIR_CTL1.DMAEN` bit enables and disables DMA on the FIR accelerator.	
		0	DMA disabled
		1	DMA enabled

Table 12: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:1 (R/W)	CH	Number of Channels. The `FIR_CTL1.CH` bit field configures the number of channels and is programmable from 0 to 31.	
		0	Channel 1
		1-30	Channel 2-31
		31	Channel 32
0 (R/W)	EN	FIR Enable. The `FIR_CTL1.EN` bit enables and disables the FIR accelerator.	
		0	Disable FIR
		1	Enable FIR

FIR Channel Control Register

The `FIR_CTL2` register is used to configure the channel specific parameters such as filter TAP length, window size, sample rate conversion, up/down sampling and ratio.

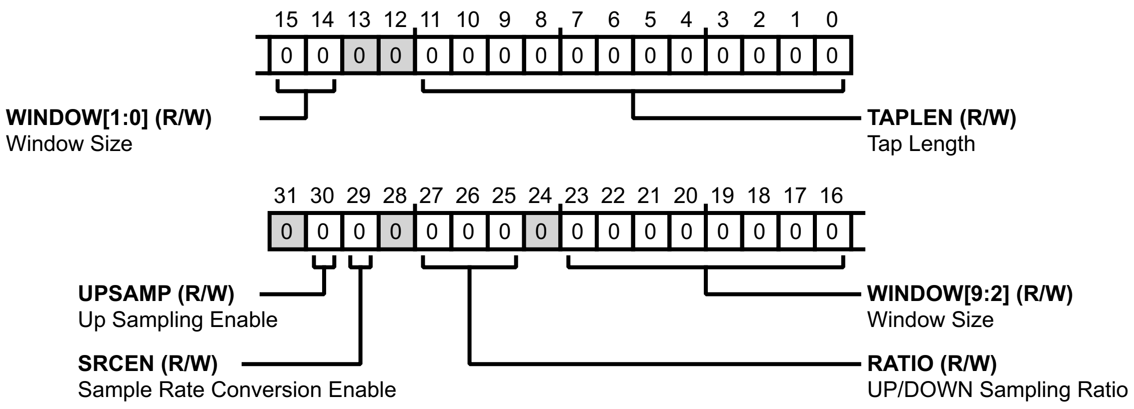

Figure 13: FIR_CTL2 Register Diagram

Table 13: FIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	UPSAMP	Up Sampling Enable. The `FIR_CTL2.UPSAMP` bit enables up sampling.	
		0	Down Sampling
		1	Up sampling

Table 13: FIR_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SRCEN	Sample Rate Conversion Enable. The `FIR_CTL2.SRCEN` bit field enables sample rate conversion.	
		0	Disabled
		1	Enabled
27:25 (R/W)	RATIO	UP/DOWN Sampling Ratio. The `FIR_CTL2.RATIO` bit field sets the sampling ratio (`FIR_CTL2.RATIO` + 1).	
23:14 (R/W)	WINDOW	Window Size. The `FIR_CTL2.WINDOW` bit field sets the window size which specifies the number of sample/block to process (sample based processing = window size of 1).	
11:0 (R/W)	TAPLEN	Tap Length. The `FIR_CTL2.TAPLEN` bit field sets the tap length which is programmable between 0-4095 (Tap Length = `FIR_CTL2.TAPLEN` + 1).	

Debug Address Register

The `FIR_DBG_ADDR` register holds the debug address. TBD

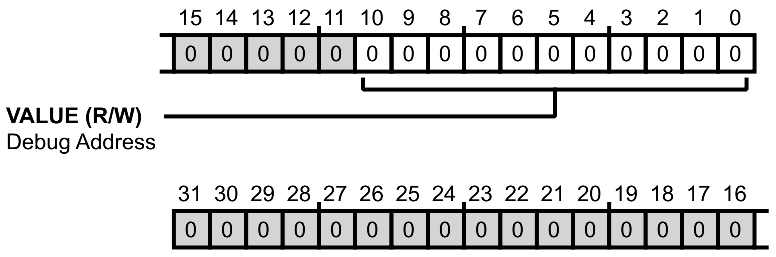

Figure 14: FIR_DBG_ADDR Register Diagram

Table 14: FIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address. The `FIR_DBG_ADDR.VALUE` bit field holds the debug address.

FIR Debug Control Register

The `FIR_DBG_CTL` register controls debugging operations such as enabling debug mode running, hold or single stepping.

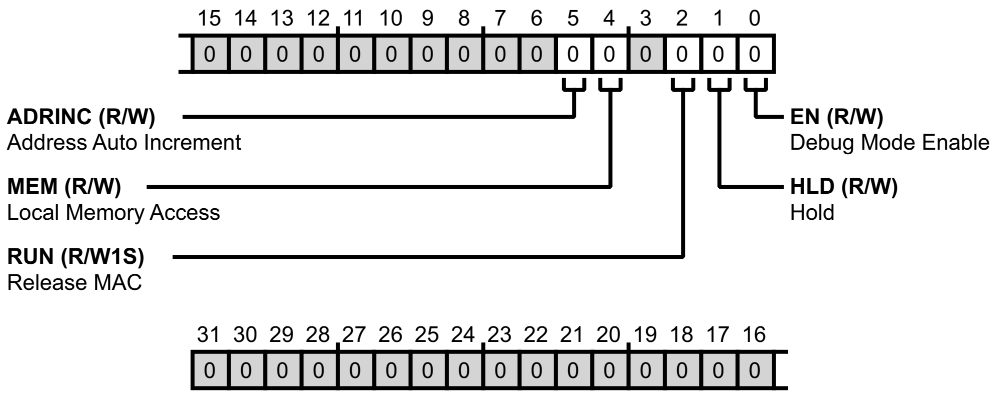

Figure 15: FIR_DBG_CTL Register Diagram

Table 15: FIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	ADRINC	Address Auto Increment. The `FIR_DBG_CTL.ADRINC` bit bit allows the address register to auto increment on `FIR_DBG_WRDAT` writes and `FIR_DBG_RDDAT` reads.	
4 (R/W)	MEM	Local Memory Access. When the `FIR_DBG_CTL.MEM` bit is set, the data and coefficients memory can be indirectly accessed.	
2 (R/W1S)	RUN	Release MAC. The `FIR_DBG_CTL.RUN` bit releases the MAC. This bit is self clearing after one FIR clock cycle.	
1 (R/W)	HLD	Hold. The `FIR_DBG_CTL.HLD` bit holds or single steps through the FIR.	
		0	Hold
		1	Single step
0 (R/W)	EN	Debug Mode Enable. The `FIR_DBG_CTL.EN` bit enables debug mode. For local memory access, the `FIR_CTL1` register can be cleared.	
		0	Disable debug mode
		1	Enable debug mode

FIR Debug Data Read Register

The `FIR_DBG_RDDAT` register hold the debug read data.

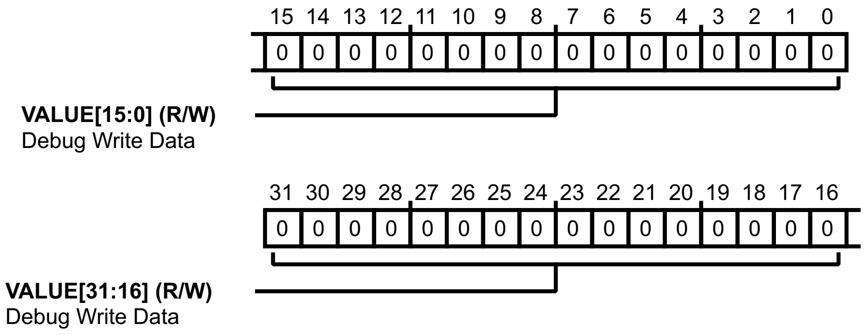

Figure 16: FIR_DBG_RDDAT Register Diagram

Table 16: FIR_DBG_RDDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data. The `FIR_DBG_RDDAT.VALUE` bit field holds the debug read data.

FIR Debug Data Write Register

The `FIR_DBG_WRDAT` register holds the debug write data.

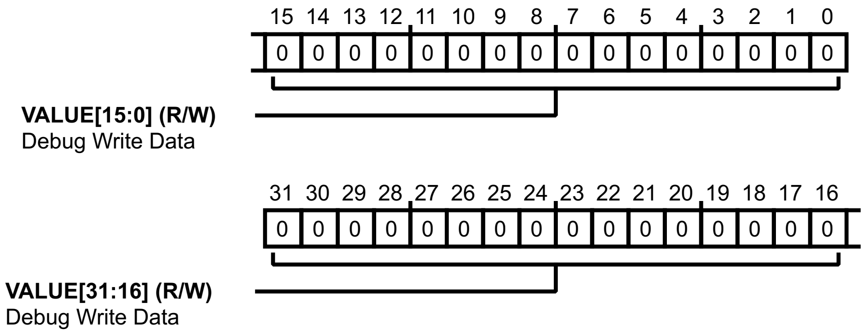

Figure 17: FIR_DBG_WRDAT Register Diagram

Table 17: FIR_DBG_WRDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data. The `FIR_DBG_WRDAT.VALUE` bit field holds the debug write data.

FIR DMA Status Register

The `FIR_DMASTAT` register provides information about chain pointer loading, coefficient DMA, data preload DMA, processing in progress, window complete, all channels complete.

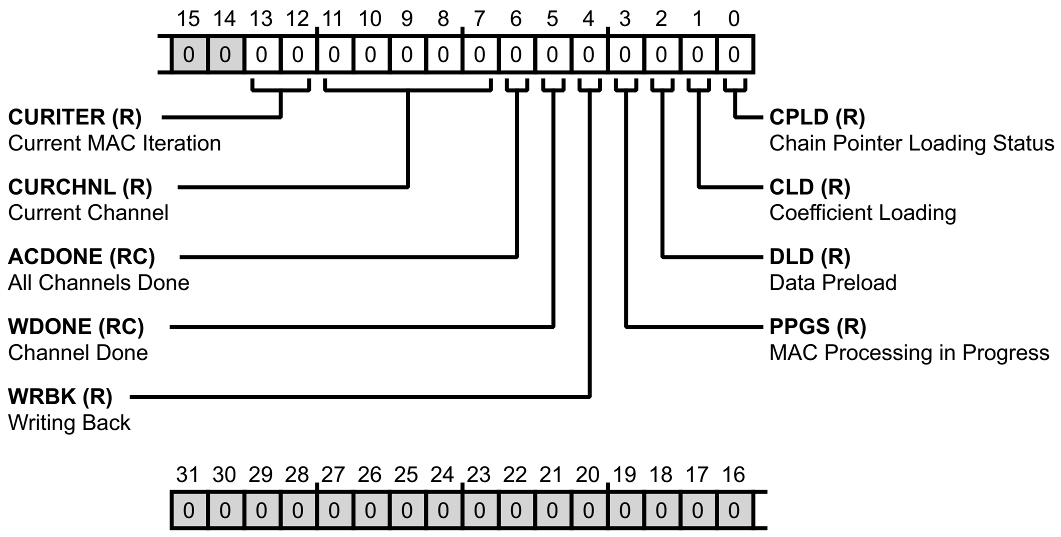

Figure 18: FIR_DMASTAT Register Diagram

Table 18: FIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:12 (R/NW)	CURITER	Current MAC Iteration. The `FIR_DMASTAT.CURITER` bit indicates the current MAC iteration in multi iteration mode. Zero indicates the final iteration.
11:7 (R/NW)	CURCHNL	Current Channel. The `FIR_DMASTAT.CURCHNL` bit indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.
6 (RC/NW)	ACDONE	All Channels Done. The `FIR_DMASTAT.ACDONE` bit indicates the accelerator that processing all channels is complete. This is a sticky bit and is cleared on a register read. The `FIR_CTL1.CCINTR` bit does not affect the `FIR_DMASTAT.ACDONE` bit.
5 (RC/NW)	WDONE	Channel Done. The `FIR_DMASTAT.WDONE` bit indicates the accelerator that processing the current channel is complete. This is a sticky bit and is cleared on a register read. The `FIR_CTL1.CCINTR` bit does not affect the `FIR_DMASTAT.WDONE` bit.
4 (R/NW)	WRBK	Writing Back. The `FIR_DMASTAT.WRBK` bit indicates the accelerator is writing back the updated index registers.

Table 18: FIR_DMASTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/NW)	PPGS	MAC Processing in Progress. The `FIR_DMASTAT.PPGS` bit indicates MAC processing in progress.
2 (R/NW)	DLD	Data Preload. The `FIR_DMASTAT.DLD` bit indicates data preloading.
1 (R/NW)	CLD	Coefficient Loading. The `FIR_DMASTAT.CLD` bit indicates coefficient loading.
0 (R/NW)	CPLD	Chain Pointer Loading Status. The `FIR_DMASTAT.CPLD` bit indicates the state machine is in chain pointer load state.

FIR Input Data Base Register

The `FIR_INBASE` register contains the input word base address with the lower two bits removed.

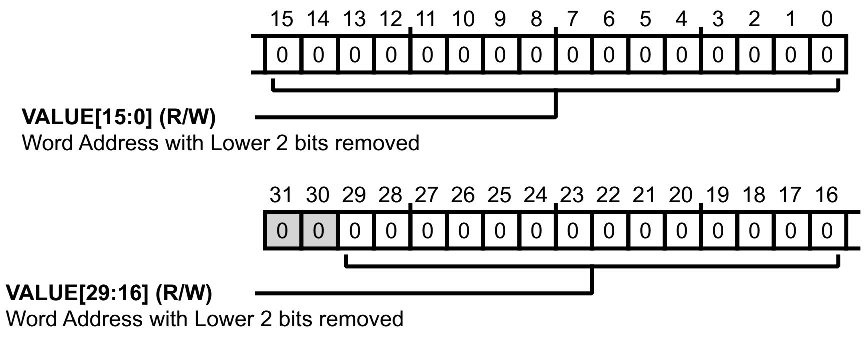

Figure 19: FIR_INBASE Register Diagram

Table 19: FIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 bits removed. The `FIR_INBASE.VALUE` bit field contains the the word address with the lower 2 bits removed.

FIR Input Data Count Register

The `FIR_INCNT` register contains the 16-bit input data count.

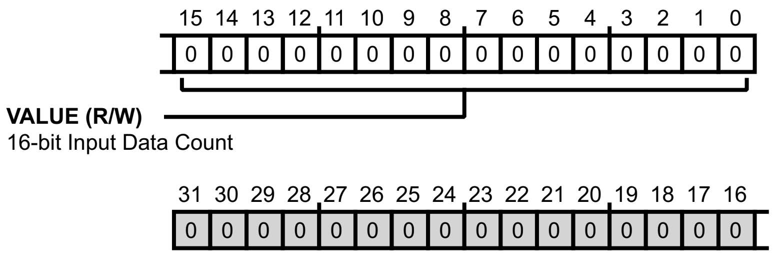

Figure 20: FIR_INCNT Register Diagram

Table 20: FIR_INCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Count. The `FIR_INCNT.VALUE` bit field contains the 16-bit input data count.

FIR Input Data Index Register

The `FIR_INIDX` register contains the input word address with the lower two bits removed.

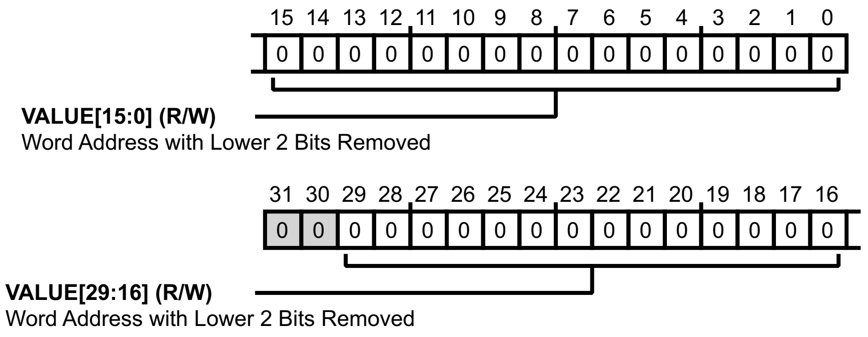

Figure 21: FIR_INIDX Register Diagram

Table 21: FIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The `FIR_INIDX.VALUE` bit field contains the input word address with the lower two bits removed.

FIR Input Data Modifier Register

The `FIR_INMOD` register contains the 16-bit input data buffer modifier.

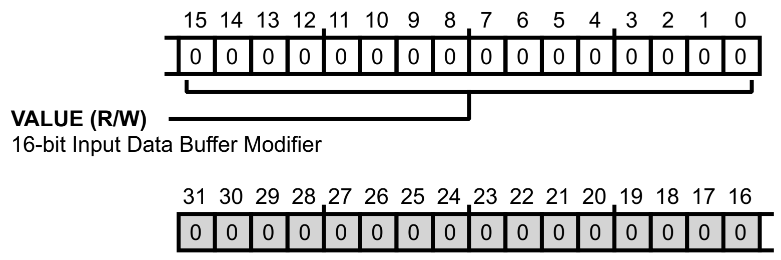

Figure 22: FIR_INMOD Register Diagram

Table 22: FIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Modifier. The `FIR_INMOD.VALUE` bit field contains the 16-bit input data buffer modifier.

FIR MAC Status Register

The `FIR_MACSTAT` register provides the status of MAC operations. The status of all four multipliers/adders are available separately for programs to poll. In fixed-point mode only the ARIx bits are used (all other bits are reserved).

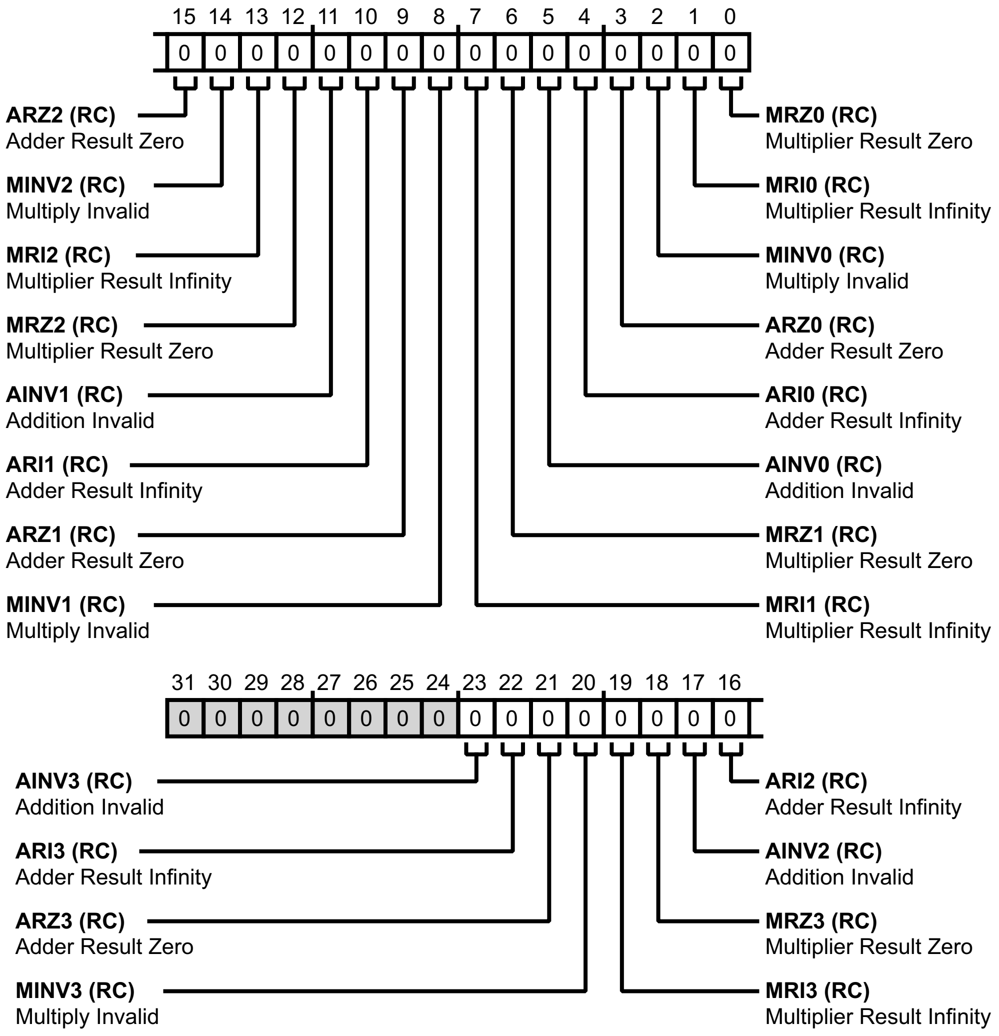

Figure 23: FIR_MACSTAT Register Diagram

Table 23: FIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23 (RC/NW)	AINV3	Addition Invalid. The `FIR_MACSTAT.AINV3` bit is set if a adder 3 addition is invalid.
22 (RC/NW)	ARI3	Adder Result Infinity. The `FIR_MACSTAT.ARI3` bit is set if adder 3 results is infinity. Indicates overflow in fixed-point mode.
21 (RC/NW)	ARZ3	Adder Result Zero. The `FIR_MACSTAT.ARZ3` bit is set if a adder 3 results is zero.

Table 23: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20 (RC/NW)	MINV3	Multiply Invalid. The `FIR_MACSTAT.MINV3` bit is set if multiplier 3 multiply operation is invalid.
19 (RC/NW)	MRI3	Multiplier Result Infinity. The `FIR_MACSTAT.MRI3` bit is set if multiplier 3 results is infinity.
18 (RC/NW)	MRZ3	Multiplier Result Zero. The `FIR_MACSTAT.MRZ3` bit is set if multiplier 3 results is zero.
17 (RC/NW)	AINV2	Addition Invalid. The `FIR_MACSTAT.AINV2` bit is set if a adder 2 addition is invalid.
16 (RC/NW)	ARI2	Adder Result Infinity. The `FIR_MACSTAT.ARI2` bit is set if adder 2 results is infinity. Indicates overflow in fixed-point mode.
15 (RC/NW)	ARZ2	Adder Result Zero. The `FIR_MACSTAT.ARZ2` bit is set if a adder 2 results is zero.
14 (RC/NW)	MINV2	Multiply Invalid. The `FIR_MACSTAT.MINV2` bit is set if multiplier 2 multiply operation is invalid.
13 (RC/NW)	MRI2	Multiplier Result Infinity. The `FIR_MACSTAT.MRI2` bit is set if multiplier 2 results is infinity.
12 (RC/NW)	MRZ2	Multiplier Result Zero. The `FIR_MACSTAT.MRZ2` bit is set if multiplier 2 results is zero.
11 (RC/NW)	AINV1	Addition Invalid. The `FIR_MACSTAT.AINV1` bit is set if a adder 1 addition is invalid.
10 (RC/NW)	ARI1	Adder Result Infinity. The `FIR_MACSTAT.ARI1` bit is set if adder 1 results is infinity. Indicates overflow in fixed-point mode.
9 (RC/NW)	ARZ1	Adder Result Zero. The `FIR_MACSTAT.ARZ1` bit is set if a adder 1 results is zero.
8 (RC/NW)	MINV1	Multiply Invalid. The `FIR_MACSTAT.MINV1` bit is set if multiplier 1 multiply operation is invalid.
7 (RC/NW)	MRI1	Multiplier Result Infinity. The `FIR_MACSTAT.MRI1` bit is set if multiplier 1 results is infinity.
6 (RC/NW)	MRZ1	Multiplier Result Zero. The `FIR_MACSTAT.MRZ1` bit is set if multiplier 1 results is zero.
5 (RC/NW)	AINV0	Addition Invalid. The `FIR_MACSTAT.AINV0` bit is set if a adder 0 addition is invalid.

Table 23: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (RC/NW)	ARI0	Adder Result Infinity. The FIR_MACSTAT.ARI0 bit is set if adder 0 results is infinity. Indicates overflow in fixed-point mode.
3 (RC/NW)	ARZ0	Adder Result Zero. The FIR_MACSTAT.ARZ0 bit is set if a adder 0 results is zero.
2 (RC/NW)	MINV0	Multiply Invalid. The FIR_MACSTAT.MINV0 bit is set if multiplier 0 multiply operation is invalid.
1 (RC/NW)	MRI0	Multiplier Result Infinity. The FIR_MACSTAT.MRI0 bit is set if multiplier 0 results is infinity.
0 (RC/NW)	MRZ0	Multiplier Result Zero. The FIR_MACSTAT.MRZ0 bit is set if multiplier 0 results is zero.

FIR Output Data Base Register

The FIR_OUTBASE register contains the output word base address with the lower two bits removed

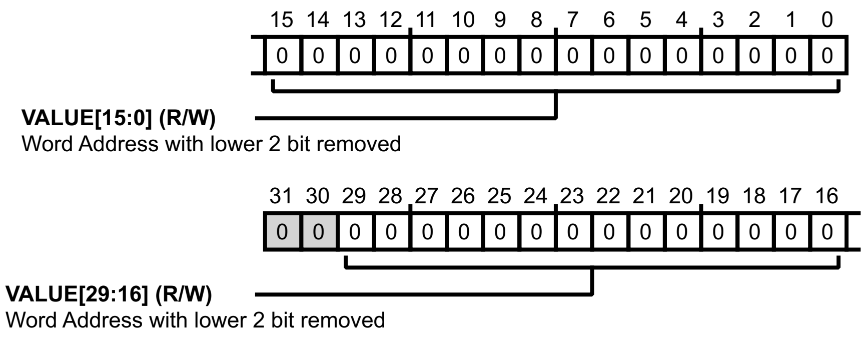

Figure 24: FIR_OUTBASE Register Diagram

Table 24: FIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with lower 2 bit removed. The FIR_OUTBASE.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Output Data Count Register

The `FIR_OUTCNT` register contains the 16-bit output buffer count.

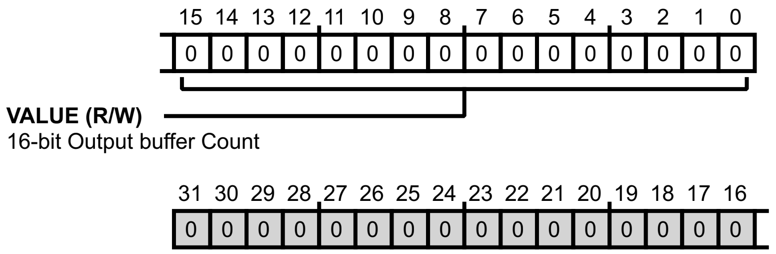

Figure 25: FIR_OUTCNT Register Diagram

Table 25: FIR_OUTCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output buffer Count. The `FIR_OUTCNT.VALUE` bit field contains the 16-bit output buffer count.

FIR Output Data Index Register

The `FIR_OUTIDX` register contains the output word address with the lower two bits removed.

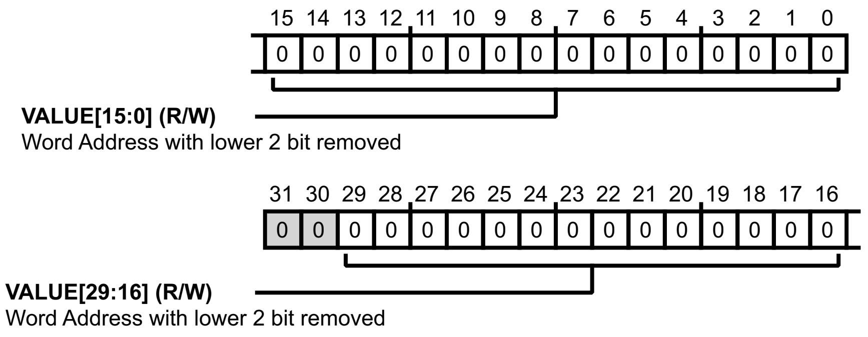

Figure 26: FIR_OUTIDX Register Diagram

Table 26: FIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with lower 2 bit removed. The `FIR_OUTIDX.VALUE` bit field contains the the word address with the lower 2 bits removed.

FIR Output Data Modifier Register

The FIR_OUTMOD register contains the 16-bit output data buffer modifier.

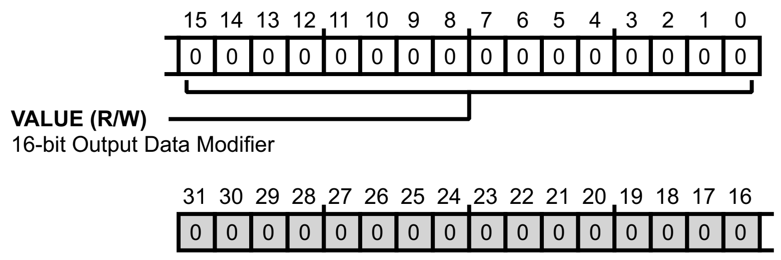

Figure 27: FIR_OUTMOD Register Diagram

Table 27: FIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Modifier. The FIR_OUTMOD.VALUE bit field contains the 16-bit output data modifier.