DA-FIR Filter Generator User's Guide

August 2010
IPUG58_01.6

Table of Contents

Introduction

This user's guide provides a description of the Distributed Arithmetic Finite Impulse Response (DA-FIR) Filter Generator IP core for the LatticeECP/EC™, Lattice ECP2/S™, LatticeECP2M/S™, LatticeECP3™, LatticeSC/M™, LatticeXP™ and LatticeXP2™ device families. The Lattice DA-FIR Filter Generator IP implements a highly configurable, multi-channel DA-FIR filter, using distributed arithmetic algorithms implemented in FPGA Look Up Tables (LUTs) or Embedded Block Memories (EBRs) to efficiently support the sum-of-product calculations required to perform the filter function. Directions for specifying the IP core configuration and including it in a design, as well as directions for simulation and synthesis, are discussed.

In addition to single rate filters, the IP core also supports a range of polyphase interpolation and decimation filters. The DA-FIR Filter IP core supports up to 32 channels, with each having up to 1024 taps. The input data, coefficient and output data widths are configurable over a wide range. The IP core uses full internal precision while allowing variable output precision with several choices for saturation and rounding.

The DA-FIR Generator IP core package comes with the following documentation and files:

- User's guide
- Protected netlist/database
- Behavioral RTL simulation model
- Source files for instantiating and evaluating the core

The DA-FIR Filter IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase on an IP license. It may also be used to evaluate the core in hardware in user-defined designs. Details for using the hardware evaluation capability are described in the Hardware Evaluation section of this document.

Quick Facts

Table 1-1 gives quick facts about the DA-FIR Filter IP core for LatticeECP2M, LatticeECP3, and LatticeSCM devices, respectively. These numbers are typical for Lattice device families that support the DA-FIR Filter IP core.

Table 1-1. DA-FIR Filter IP Core Quick Facts

		DA-FIR IP Configuration					
		Parallel 32-tap Filter		**Serial 16-tap Filter**		**Parallel 32-tap Filter**	**Serial 16-tap Filter**
Core Requirements	FPGA Families Supported	Lattice ECP2M and Lattice ECP3				Lattice SCM	
	Minimal Device Needed	LFE2M20E	LFE3-17E	LFE2M20E	LFE3-17E	LFSCM3GA15EP1	LFSCM3GA15EP1
Resource Utilization	Targeted Device	LFE2M20E-6F256C	LFE3-70E-7FN484CES	LFE2M20E-6F256C	LFE3-70E-7FN484CES	LFSCM3GA15EP1-6F256C	LFSCM3GA15EP1-6F256C
	Data Path Width	16	16	16	16	16	16
	LUTs	5097	5068	376	348	5021	345
	sysMEM EBRs	0	0	0	0	0	0
	Registers	5979	5950	481	476	6015	481
Design Tool Support	Lattice Implementation	Diamond® 1.0 or ispLEVER® 8.1					
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2009.12L-1					
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition					
		Mentor Graphics® ModelSim™ SE 6.3F					

Features

- Variable number of taps up to 1024
- Multi-channel support (up to 32 channels)
- Polyphase interpolation/decimation filters
- Halfband filters
- Interpolation and Decimation ratios from 2 to 32
- Input data widths from 4 to 32 bits
- Coefficient widths from 4 to 32 bits
- Signed or unsigned data and coefficients
- Selectable rounding: truncation, rounding away from zero, convergent rounding
- Optional saturation logic for overflow handling
- Full precision arithmetic
- Specification of fractional inputs and outputs
- Support for both serial and parallel filters, with user specified degree of parallelism.
- Configurable pipelining to increase performance
- Optimizations based on filter characteristics (symmetry and halfband).
- Handshake signals to facilitate smooth interfacing

This chapter provides a functional description of the DA-FIR Filter IP core.

General Description

Distributed Arithmetic is a computation algorithm that performs multiplication with look up tables and shift and add functions instead of multipliers. A generalized Finite Impulse Response (FIR) filter performs a convolution of its input data sequence with its impulse response (the discrete-time inverse Fourier transform of its desired frequency response). The equation for performing the convolution is given by

(1)

$$y(n) = \sum_{k=0}^{K-1} x_k(n) A_k$$

where y(n) is the filter output, the variables xk(n) are the input samples, and the values Ak are the filter coefficients. The equation implies that a single output response is the sum of K product terms. In the DA filter the product terms are generated by look up tables instead of multipliers.

Equation 1 is rewritten for a single output using 2's complement representation of xk, where xkb is a binary variable and can assume only values of 0 and 1. The sign bit with value -1 is indicated by xk0. B is the input data width.

(2)

$$y = \sum_{k=0}^{K-1} A_k(-x_{k0} + \sum_{b=1}^{B-1} x_{kb} 2^{-b})$$

$$= -\sum_{k=0}^{K-1} A_k x_{k0} + \sum_{k=0}^{K-1} \sum_{b=1}^{B-1} A_k x_{kb} 2^{-b}$$

Explicitly expressing all the product terms under the summation symbols

$$y = -[x_{00} \bullet A_0 + x_{10} \bullet A_1 + ... + x_{(K-1)0} \bullet A_{K-1}]$$

$$+ [x_{01} \bullet A_0 + x_{11} \bullet A_1 + ... + x_{(K-1)1} \bullet A_{K-1}]2^{-1}$$

$$+ [x_{02} \bullet A_0 + x_{12} \bullet A_1 + ... + x_{(K-1)2} \bullet A_{K-1}]2^{-2}$$

$$\bullet$$
$$\bullet$$
$$\bullet$$

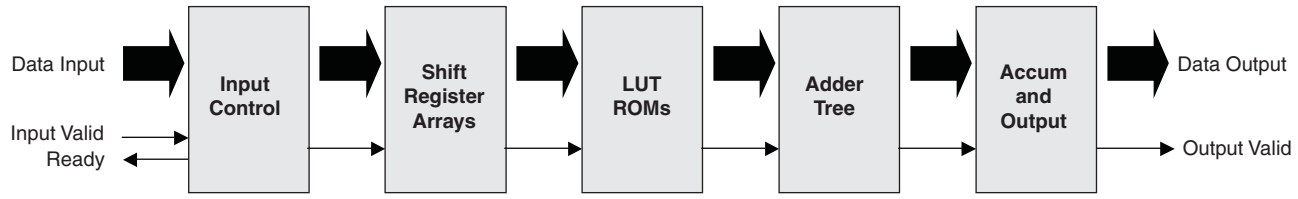
$$+ [x_{0(B-2)} \bullet A_0 + x_{1(B-2)} \bullet A_1 + ... + x_{(K-1)(B-2)} \bullet A_{K-1}]2^{-(B-2)}$$

$$+ [x_{0(B-1)} \bullet A_0 + x_{1(B-1)} \bullet A_1 + ... + x_{(K-1)(B-1)} \bullet A_{K-1}]2^{-(B-1)}$$

Each product term within the square brackets is a binary AND operation involving a bit of the input variable and all bits of the constant. The plus signs denote arithmetic sum operations. The exponential factors denote scaled contributions of the bracketed partial sums to the total sum. The look-up table can be addressed by the same bit of all input variables and provides the sum of the terms within each pair of square brackets.

Functional Block Diagram

Figure 2-1. DA-FIR Filter Generator Block Diagram



DA-FIR Filter IP Core I/O

Figure 2-2 shows the I/O interface view of the DA-FIR Filter IP core.

Figure 2-2. DA-FIR Filter IP Core I/O

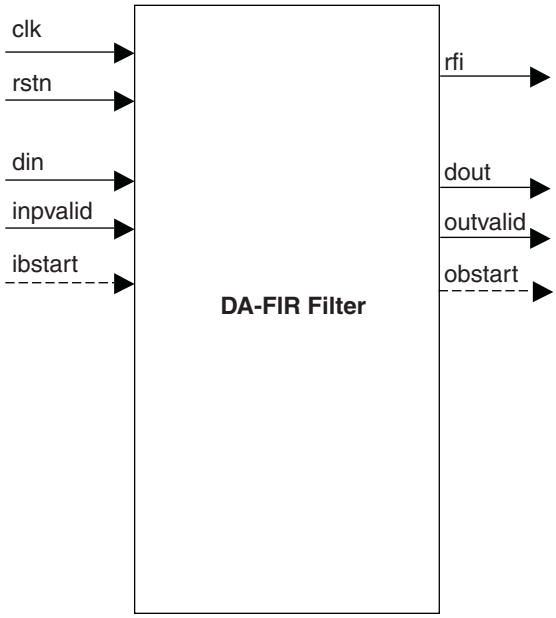


Table 2-1 defines all I/O interface ports available in this core.

Table 2-1. Top-Level Port Definitions

Port	Bits	I/O	Description
General I/Os			
clk	1	I	System clock for data and control inputs and outputs.
rstn	1	I	System wide asynchronous active-low reset signal.
din	Input data width	I	Input data.
inpvalid	1	I	Input valid signal. The input data is read-in only when inpvalid is high.
dout	Output width	O	Output data
outvalid	1	O	Output data qualifier. Output data dout is valid only when this signal is high.

Table 2-1. Top-Level Port Definitions (Continued)

Port	Bits	I/O	Description
rfi	1	O	Ready for input. This output, when high, indicates that the IP core is ready to receive the next input data. A valid data may be applied at din only if rfi was high during the previous clock cycle.
When Number of Channels is Greater than 1			
ibstart	1	I	Input block start. For multi-channel configurations, this input identifies channel 0 of the input.
obstart	1	O	Output block start. For multi-channel configurations, this output identifies channel 0.

Data Path

Adder Tree

The Adder Tree is implemented as a pipelined adder. Each input word is split into slices, and a tree of adders computes the sum for each slice (including carry bits). A second stage then handles carry propagation. The output is a full precision sum.

The number of slices per word is set based on the Performance parameter.

Accumulator

The Accumulator computes a running sum of its input values (the output of the Adder Tree). The Accumulator is pipelined and the size and number of slices is determined by the parameter, Performance.

The Accumulator output is large enough to accommodate the sum of all tap-coefficient products for the longest filter supported by the specified configuration.

Rounding

The Rounding module provides three types of rounding, depending on the parameter, Rounding as:

- None (truncation) – Discards all bits to the right of the output least significant bit and leaves the output uncorrected.
- Rounding away from zero – Rounds away from zero if the fractional part is exactly one-half.
- Convergent rounding – Rounds to the nearest even value if the fractional part is exactly one-half.

Output

The DA-FIR Filter IP core provides a single clock-cycle output ready signal (outvalid) to indicate valid data at dout. The value at dout is undefined when outvalid is not asserted (dout is the direct output of the DA-FIR's accumulator, so it may change with each clock cycle). For output widths less than full precision, which accumulator bits are steered to dout is determined by the output width parameter and a function of the input and output binary point values The values of these parameters are set in the GUI provided in the IPExpress™ tool.

Interpolation and Decimation

The interpolation process effectively inserts a specified number of zero-value samples in between input data samples and smooths the output with low-pass filtering.

The output is a filtered version of the input at a rate equal to the input data rate multiplied by the specified interpolation factor. The DA-FIR employs the polyphase interpolating structure by dividing the coefficient set into multiple phases.

Decimation provides a down-sampled version of the filtered signal at a rate equal to the input rate divided by the decimation factor. The DA-FIR employs the polyphase decimating structure by dividing the coefficient set into multiple phases.

Throughput Considerations

The rate at which the DA-FIR Filter IP core can accept data is determined by the number of clock cycles required to perform each convolution, which depends on input data width (DINWIDTH) and the number of bits processed per clock cycle (or Bits processed at a time- BAAT) as shown below:

For non-symmetric single rate and interpolation FIR:

- Clocks_per_input = DINWITH/BAAT

For symmetric and halfband single rate and interpolation FIR:

- Clocks_per_input = (DINWIDTH+1)/BAAT

For non-symmetric decimation FIR:

- Clocks_per_output = DINWITH/BAAT

For symmetric and halfband decimation FIR:

- Clocks_per_output = (DINWIDTH+1)/BAAT

Coefficient Memory

The filter coefficients are stored in the internal ROM. A valid coefficient set for a selected filter architecture must be provided to the DA-FIR IP in order for proper filter operation. For example, a generated symmetric DA-FIR requires that the coefficient set is also symmetric. The coefficients are then used to process the filter operations for all channels.

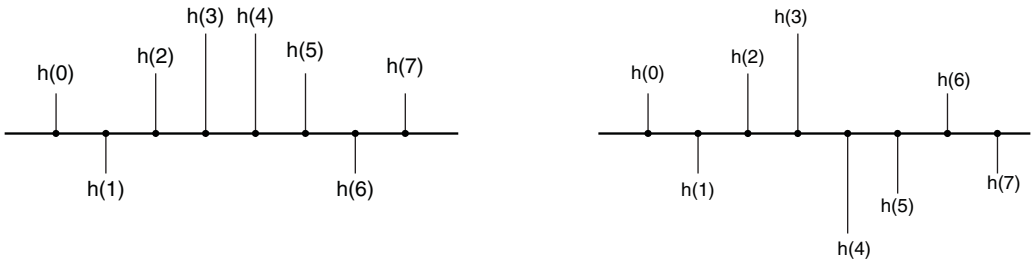
The DA-FIR IP supports the ROM generation using either distributed or EBR-based memory. The DA-FIR Filter automatically generates the appropriate memory structure, but neither the GUI nor the compiler software ensures that the selected device has sufficient memory resources. Choosing distributed memory provides a finer grained memory system, but utilizes more FPGA logic resources.

Architecture Optimizations

The DA-FIR Filter IP core supports the following architecture optimizations depending on the characteristics of the filter.

Many DSP applications utilize symmetric FIR configurations. This symmetry can be exploited in the DA-FIR architecture to minimize the number of LUTs required to achieve the same throughput. Figure 2-3 shows examples of positive and negative symmetry filter coefficients. For the positive symmetry example shown in Figure 2-3: coefficients h(0) = h(7), h(1) = h(6), h(2) = h(5), and h(3) = h(4). For the negative symmetry example shown in Figure 2-3: coefficients h(0) = -h(7), h(1) = -h(6), h(2) = -h(5), and h(3) = -h(4). The symmetry architecture optimization can be applied to all filters.

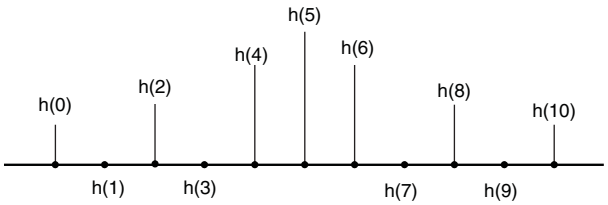
Figure 2-3. Impulse Response of Symmetric FIR



Impulse Response of Positive Symmetric **Impulse Response of Negative Symmetric**

Figure 2-4 shows the impulse response of a halfband FIR filter. The half band filter has odd number of coefficients with the alternate coefficients being zero, except for the middle one. When this impulse response is used in a filter that interpolates or decimates by a factor of two, all the zero coefficients fall into one phase, and that phase will reduce to only the center coefficient. The halfband polyphase optimization is only intended for interpolated or decimated filter by a factor of 2.

Figure 2-4. Impulse Response of Halfband FIR



Impulse Response of Halfband Filter

Configuring the DA-FIR Filter IP core

Architecture Options

The options for number of channels, number of taps and filter type are independent of each other and are directly specified in the “Architecture” tab of the GUI. If a polyphase decimator or interpolator is required, the decimation or interpolation factor can be directly specified in the GUI.

Coefficients Specification

The coefficients of the filter are specified using a coefficients file. The coefficients file is a text file with one coefficient per line. If the coefficients are symmetric, the check box “Symmetric Coefficients” has to be checked so the IP core uses symmetry adders to reduce the number of effective taps. If the symmetric coefficients box is checked, only the coefficients corresponding to one half of the number of taps are read from the coefficient file. For an n tap symmetric coefficients filter, the number of coefficients read from the coefficients file is equal to ceil(n/2).

The coefficient values in the file can be in any radix selected by the user- floating point, decimal, hexadecimal or binary. A unary negative operator is used only if the coefficients are specified in floating point or decimal radix. For hexadecimal and binary radices, the numbers must be represented in twos complement form. For decimal, hexadecimal and binary radices, the numbers must be an integer; If you have floating point coefficients, you should convert them to integer by the formula: Ak_c = round (Ak * pow(2,CPOINTS)), where CPOINTS is the coefficients binary point position.

An example coefficients file in decimal format for 11 taps, 16 bits coefficients set is given below.

-556
-706
-857
-419
1424
5309
11275
18547
25649
30848
32758

An example coefficients file in floating point format for the above case when the Coefficients binary point position is 8, is given below. The coefficients will be quantized to conform to the 16.8 fractional number.

-2.1719
-2.7578
-3.3477
-1.6367
5.5625
20.7383
44.043
72.45
100.0191
120.5
127.96

Throughput

The throughput and the resource utilization can be controlled by assigning a proper value to “Number of bits processed per clock cycle” (or BAAT) parameter. The BAAT value should be a factor of DINWIDTH for non-symmetric FIR and DINWIDTH+1 for symmetric and half band FIR configurations, where DINWIDTH is input data width. For example, if DINWIDTH is 18, the allowable BAAT values are 1,2,3,6,9 and 18.

For a single rate filter, the highest throughput of one input per clock cycle can be achieved by setting the BAAT to DINWIDTH. For single-rate symmetric or half-band configurations, the highest throughput of one input per clock cycle is obtained by setting BAAT to (DINWIDTH+1). To reduce the resource utilization at the expense of reduced throughput, a lower value of BAAT may be specified.

The interpolator gives out multiple output data (equal to Interpolation factor or IFACTOR) for each input data. The maximum throughput is obtained when the input data is applied once every IFACTOR clock cycles. The ideal BAAT for maximum throughput is equal to ceil (DINWIDTH/IFACTOR). However this ideal BAAT value may not be selectable, if it is not a factor of DINWIDTH. A higher BAAT value can be selected, but it would mean more resources that are not fully utilized. In such cases, the input data width may be increased to allow the selection of the ideal BAAT value. If the input data width is set higher than the actual input data width, the inputs need to be sign extended accordingly. As an example, if DINWIDTH is 20, the selectable BAAT values are 1,2,4,5,10 and 20. If IFACTOR is 3, the ideal BAAT value is ceil(20/3)=7. For maximum throughput, the BAAT value of 10 has to be chosen, as the ideal value 7 is not available. In this case, the input width may be set to 21 and input data sign extended by 1 bit to allow for the selection of BAAT values- 1,2,3,7 and 21. The ideal BAAT value of 7 can now be selected. For symmetric or half band interpolators, (DINWIDTH+1) is used in place DINWIDTH for determining the ideal BAAT value.

The decimator reads more input data samples (equal to Decimation factor or DFACTOR) for each output data. All the DFACTOR input data samples are processed in parallel to obtain the output sample. The ideal value for BAAT for maximum throughput is equal to ceil (DINWIDTH/DFACTOR). If the ideal BAAT value is not selectable, the input data width may be increased to allow the selection of the ideal or a slightly higher BAAT value. If the input data width is set higher than the actual input data width, the inputs need to be sign extended accordingly. As an example, if DINWIDTH is 22, the selectable BAAT values are 1,2,11 and 22. If DFACTOR is 4, the ideal BAAT value is ceil(22/4)=6. For maximum throughput, the BAAT value of 11 has to be chosen, as the ideal value 6 is not available. In this case, the input width may be set to 24 and input data sign extended by 2 bits to allow for the selection of BAAT values- 1,2,3,4,6,8,12 and 24. The ideal BAAT value of 6 can now be selected. For symmetric or half band decimators, (DINWIDTH+1) is used in place of DINWIDTH for determining the ideal BAAT value.

I/O Specification Options

The controls in this tab are used to define the various widths and precision methods in the data path. The width and binary point positions of the input data and coefficients can be defined independently. From the input data width, coefficient width and the number of taps, the full precision output width and true location of the output binary point automatically get fixed. The full precision output is converted to user specified output width by dropping some least

significant (LS) and some most significant (MS) bits and by performing the specified rounding and overflow processing. The output is specified by the output width and the output binary point position parameter.

Implementation Options

Options in this box are used to specify memory type, optimization level and synthesis options.

Memory Type

For multi-channel interpolator, there is an output buffer for reordering the output data. The depth of the output buffer is (1<<(log2(NUM_CHAN * IFACTOR)+1)), where NUM_CHAN is the number of channels.

For decimator, there is an input buffer for reordering the input samples. The depth of the output buffer is (1<<(log2(NUM_CHAN * DFACTOR)+1)).

Coefficient memory type, input buffer type and output buffer type can be configured to either distributed or EBR-based memory.

Performance

The performance optimization level can be specified from 1 to 4. This number determines how the data word is sliced in the adder trees and accumulators to cut down the word size of the adders and thus improve performance. In level 1, there is no word splitting or pipeline registers between adder stages. In level 2, there is no word splitting, but there is a pipeline register after each stage of the adder trees or accumulators. Each input word to the adder or accumulator is split into two slices when the performance is set to level 3 and four slices when set to level 4. Due to the word splitting and pipelining, the resource utilization as well as the maximum frequency of operation increases when the performance level goes up. However, there may not be a performance increase for lower data widths and/or smaller number of taps even when the performance level is set to a higher value.

The default optimization level is 2.

Synthesis Options

The synthesis options, Frequency constraint and Pipelining and retiming are passed on to the synthesis tool when the IP is generated. “Pipelining and retiming” option is used to move existing registers to balance delays between registers. Users can adjust these two options to get a better result.

Interfacing with the DA-FIR Filter IP core

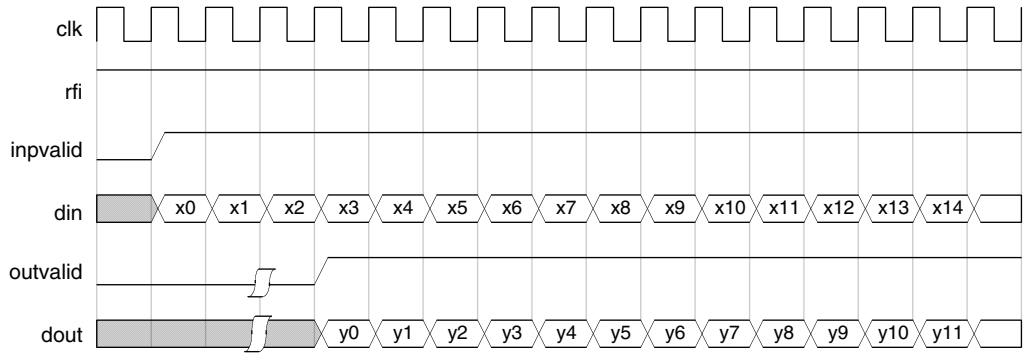
For multi-channel implementations, two ports, ibstart and obstart, are available in the IP core to synchronize the channel numbers. The input ibstart is used to identify channel 0 data applied at the inputs. The output obstart goes high simultaneously with channel 0 output data.

Timing Description

Timing diagrams for the DA-FIR Filter IP core are given in Figure 2-5 through Figure 2-9.

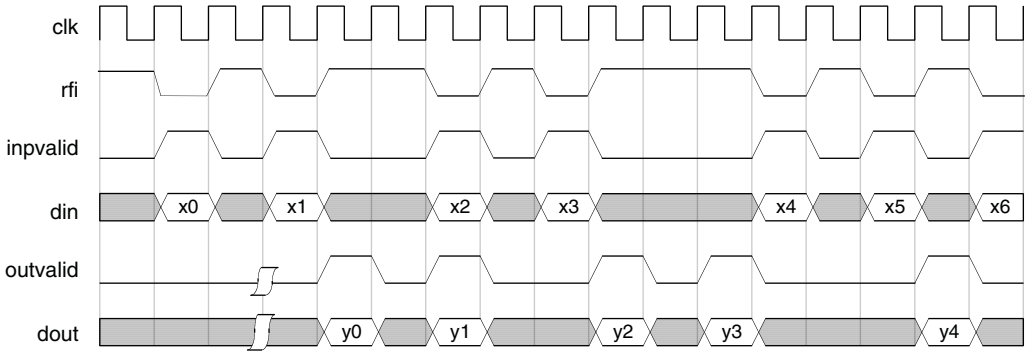
Figure 2-5 shows the case where BAAT is equal to DINWIDTH. The IP core can accept and process the input data continuously. x0, x1, x2… are input data and y0, y1, y2…are the output data.

Figure 2-5. Timing Diagram for a Single Channel, Single Rate FIR Filter with Continuous Inputs



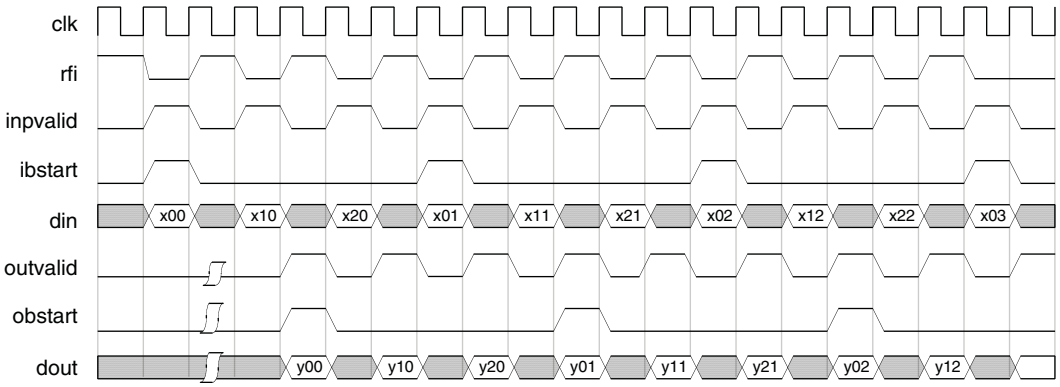
In Figure 2-6, BAAT is equal to ceil (DINWIDTH/2). The IP core can accept and process one input data every two clock cycles.

Figure 2-6. Timing Diagram for a Single Channel, Single Rate FIR Filter with Gaps in Input



In Figure 2-7, BAAT is equal to (DINWIDTH/2) and the number of channels is 3.The IP core can accept and process one input data for every two clock cycles. In this and following figures, x00, x01, x02… are the input data of the channel 0; x10, x11, x12… are the input data of the channel 1; x20, x21, x22… are the input data for channel 2. Similarly, y00, y01, y02… are the output data for channel 0; y10, y11, y12… are the output data for channel 1 and y20, y21, y22… are the output data for channel 2.

Figure 2-7. Timing Diagram for a Multi-Channel (3 Channels), Single Rate FIR Filter



In Figure 2-8, BAAT is equal to ceil (DINWIDTH/3). The IP core can accept and process one input data every three clock cycles and generate one output data every one clock cycle.

Figure 2-8. Timing Diagram for a Multi-Channel (3 Channels) Interpolator (Factor of 3)

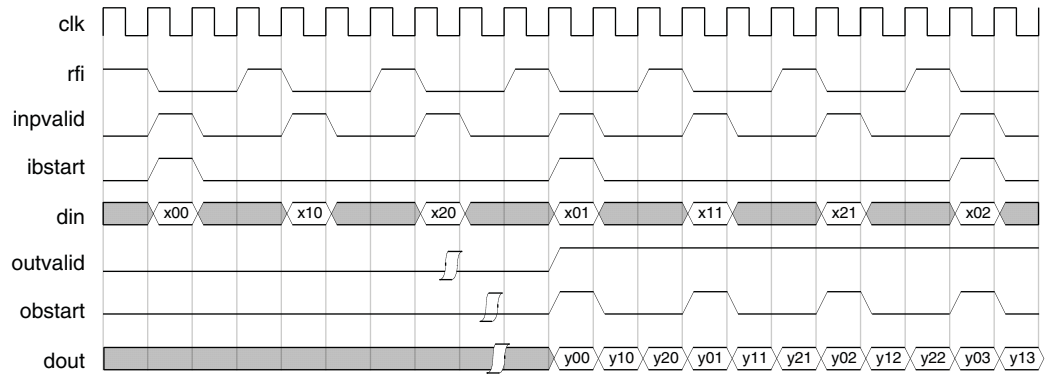
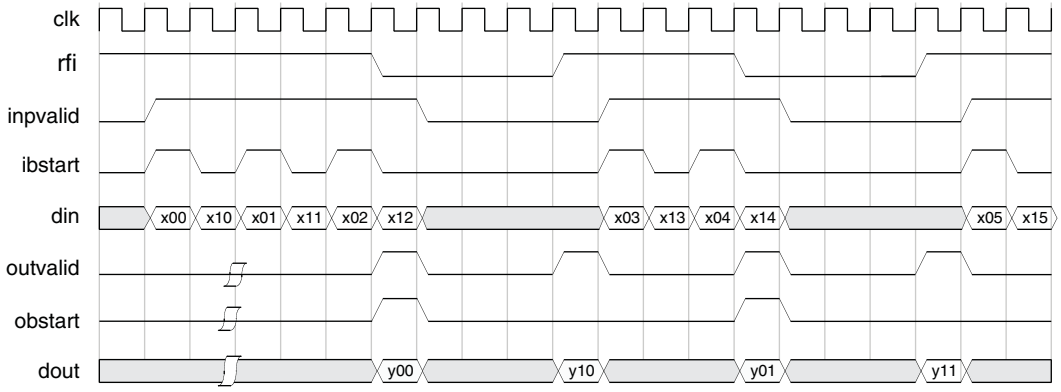


Figure 2-9 shows the 2-channel, decimation by 2 configuration where BAAT is equal to ceil (DINWIDTH/4). In this configuration, the IP core can accept and process four input data every eight clock cycles and generate two output data every eight clock cycles.

Figure 2-9. Timing Diagram for a Multi-Channel (2 Channels) Decimator (Factor of 2)



Parameter Settings

The IPexpress tool is used to create IP and architectural modules in the Diamond and ispLEVER software. Refer to "IP Core Generation" on page 21 for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the DA-FIR Filter IP core. The parameter settings are specified using the DA-FIR Filter IP core Configuration GUI in IPexpress.

Table 3-1. Parameter Specifications

Parameters	Range/Options	Default Values
Filter Specifications		
Number of channels	1 to 32	1
Number of taps	1 to 1024	63
Filter type	Single rate, Interpolator, Decimator	Single rate
Interpolation factor	2 to 32	2
Decimation factor	2 to 32	2
Coefficients Specifications		
Symmetric coefficients	Yes/No	No
Negative symmetry	Yes/No	No
Half band	Yes/No	No
Coefficient radix	Floating point, Decimal, Hex, Binary	Floating point
Coefficients file	Browse	-
Throughput		
Number of bits processed per clock cycle	1 to Input data width	1
I/O Specifications		
Input data type	Signed/Unsigned	Signed
Input data width	4 to 32	16
Input data binary point position	-2 to Input data width + 2	0
Coefficients type	Signed/Unsigned	Signed
Coefficients width	4 to 32	16
Coefficients binary point position	-2 to Coefficients width + 2	0
Output width	4 to Max Output Width	Input data width
Output binary point position	(4+Input data binary point position + coefficient binary point position - Max output width) to (Output width + Input data binary point position + Coefficient binary point position - 4)	0
Precision Control		
Overflow	Saturation/Wrap-around	Saturation
Rounding	None/Round away from zero/Convergent rounding	None
Memory Type		
Coefficient memory type	EBR/Distributed	Distributed
Input buffer type	EBR/Distributed	Distributed
Output buffer type	EBR/Distributed	Distributed
Performance		
Performance	1 to 4	2

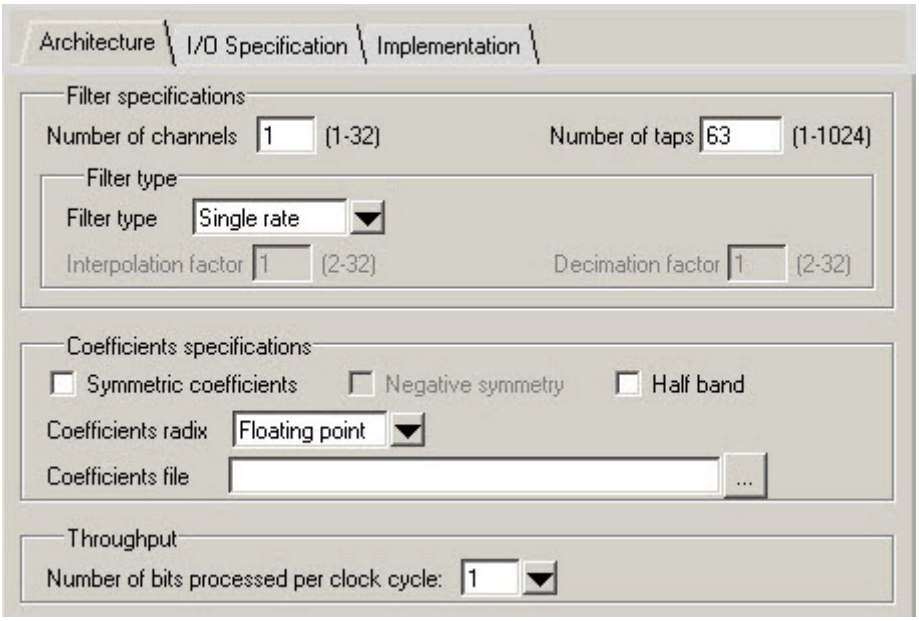
Table 3-1. Parameter Specifications (Continued)

Parameters	Range/Options	Default Values
Synthesis Options		
Frequency constraint	1-400	250
Pipelining and retiming	Yes/No	No

Architecture Tab

Figure 3-1 shows the contents of the Architecture tab.

Figure 3-1. Architecture Tab



Filter Specifications

Number of Channels

This parameter specifies the number of channels.

Number of Taps

This parameter specifies the number of taps.

Filter Type

This parameter specifies whether the filter is a single rate filter, interpolator or decimator.

Interpolation Factor

The value of the interpolation factor is displayed in this box.

Decimation Factor

The value of the decimation factor is displayed in this box.

Coefficients Specifications

Symmetric Coefficients

This parameter specifies whether the coefficients are symmetric. If this is checked only one-half of the number of coefficients (if number of taps is odd, the half value is rounded to the next higher integer) is read from the initialization file.

Negative Symmetry

If this is checked, the coefficients are considered to be negative symmetric. That is, the second half of the coefficients are made equal to the negative of the corresponding first-half coefficients.

Half Band

Specifies whether a half band filter is realized. If this is checked only one half of the number of coefficients (if the number of taps is odd, the half value is rounded to the next higher integer) is read from the initialization file.

Coefficients Radix

This parameter specifies the radix for the coefficients in the coefficients file. For decimal radix, the negative values have a preceding unary minus sign. For hexadecimal (Hex) and binary radices, the negative values must be written in two's complement form using exactly as many digits as specified by the coefficients width parameter. The floating point coefficients are specified in the form <nn...n>.<dd...d>, where the digits 'n' denote the integer part and the digits 'd', the decimal part. The values of the floating point coefficients must be consistent with the Coefficients width and Coefficients binary point position parameters.

Coefficients File

This parameter specifies the name and location of the coefficients file. If the coefficients file is not specified, the filter is initialized with a default coefficient set.

Throughput

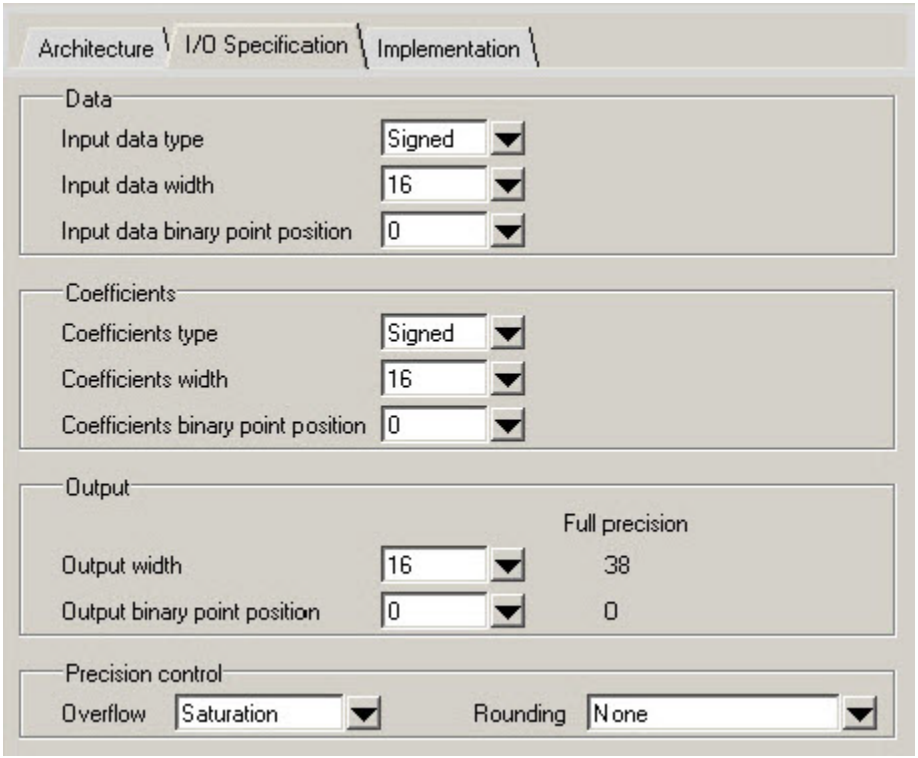
Number of Bits Processed Per Clock Cycle

This parameter specifies the number of bit processed at a time. This defines the degree of parallelism. This value must be a factor of Input data width for non-symmetric architecture, or Input data width+1 for symmetric configurations.

I/O Specification Tab

Figure 3-1 shows the contents of the I/O Specification tab.

Figure 3-2. I/O Specification



Data

Input Data Type

This parameter specifies the input data type as signed or unsigned. If the type is signed, the data is interpreted as a two's complement number.

Input Data Width

This parameter specifies the input data width.

Input Data Binary Point Position

This parameter specifies the location of the binary point in the input data. This number specifies the bit position of the binary point from the LSB of the input data. If the number is zero, the point is right after LSB, if positive, it is to the left of LSB and if negative, it is to the right of LSB.

Coefficients

Coefficients Type

This parameter specifies the coefficients type as signed or unsigned. If the type is signed, the coefficient data is interpreted as a two's complement number.

Coefficients Width

This parameter specifies coefficients width.

Coefficients Binary Point Position
This parameter specifies the location of the binary point in the coefficients. This number specifies the bit position of the binary point from the LSB of the coefficients. If the number is zero, the point is right after LSB, if positive, it is to the left of LSB and if negative, it is to the right of LSB.

Output

Output Width
The maximum full precision output width is defined by Max Output Width = Input data width + Coefficients width +ceil(Log2(Number of taps)). The core's output is usually a part of the full precision output equal to the Output width and extracted based on the different binary point position parameters. The operator ceil (x) returns the next higher integer if x is a fraction and returns the same value, if x is an integer.

The format for the internal full precision output is displayed as a static text next to the Output width control in the GUI. The format is displayed as W.F, where W is the full precision output width and F is the location of the binary point from the LSB of the full precision output, counted to the left.

Output Binary Point Position
This number specifies the bit position of the binary point from the LSB of the actual core output. If the number is zero, the point is right after LSB, if positive, it is to the left of LSB and if negative, it is to the right of LSB. This number, together with the parameter Output width determines how the actual core output is extracted from the true full precision output. The precision control parameters Overflow and Rounding are applied respectively when MSBs and LSBs are discarded from the true full precision output.

Precision Control

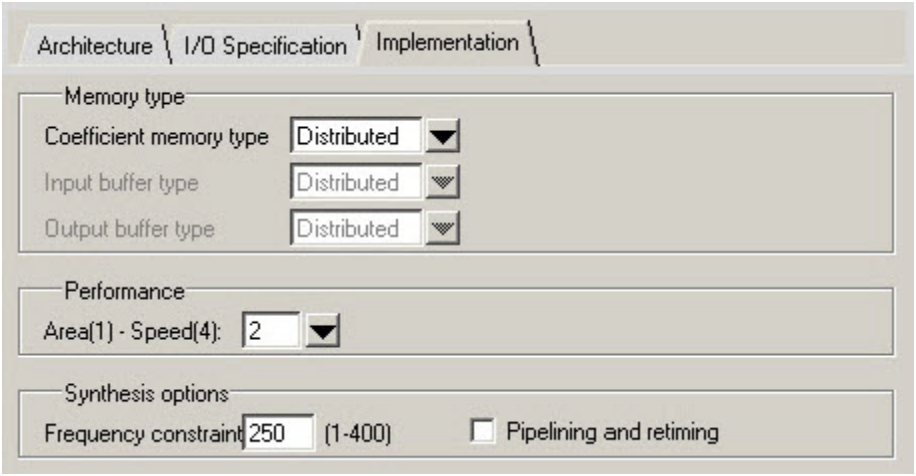
Overflow
This parameter specifies what kind of overflow control is to be used. This parameter is available whenever there is a need to drop some of the MSBs from the true output. If the selection is "Saturation", the output value is clipped to the maximum, if positive or minimum, if negative, while discarding the MSBs. If the selection is "Wrap-around", the MSBs are simply discarded without making any correction.

Rounding
This parameter specifies the rounding method when there is a need to drop one or more LSBs from the true output.

I/O Specification Tab

Figure 3-1 shows the contents of the I/O Specification tab.

Figure 3-3. I/O Specification



Memory Type

Coefficient Memory Type
This parameter selects the type of memory that is used for storing the coefficients. If the selection is “EBR”, Embedded Block RAM memories are used for storing the coefficients. If the selection is “Distributed”, distributed memories are used for storing coefficients

Input Buffer Type
This parameter specifies the memory type for the input buffer.

Output Buffer Type
This parameter specifies the memory type for the output buffer.

Performance

This section specifies the performance-resource trade-off optimization level. Lower number optimizes for area, while higher numbers results in better performance.

Synthesis Options

Frequency Constraint
This section specifies the frequency constraint that is passed on to the synthesis tool.

Pipelining and Retiming
This section specifies the pipelining and retiming synthesis options that are passed on to the synthesis tool.

IP Core Generation

This chapter provides information on licensing the DA-FIR Filter IP core, generating the core using the Diamond or ispLEVER software IPexpress tool, running functional simulation, and including the core in a top-level design. The Lattice DA-FIR Filter IP core can be used in LatticeECP3, LatticeECP2/M, LatticeXP/XP2 and LatticeSC/M device families.

Licensing the IP Core

An IP license that specifies the IP core and device family (ECP3, ECP2/M, ECP, XP2, XP or SC/M) is required to enable full use of the DA-FIR Filter IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The DA-FIR Filter IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license (see "Hardware Evaluation" on page 27 for further details). However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

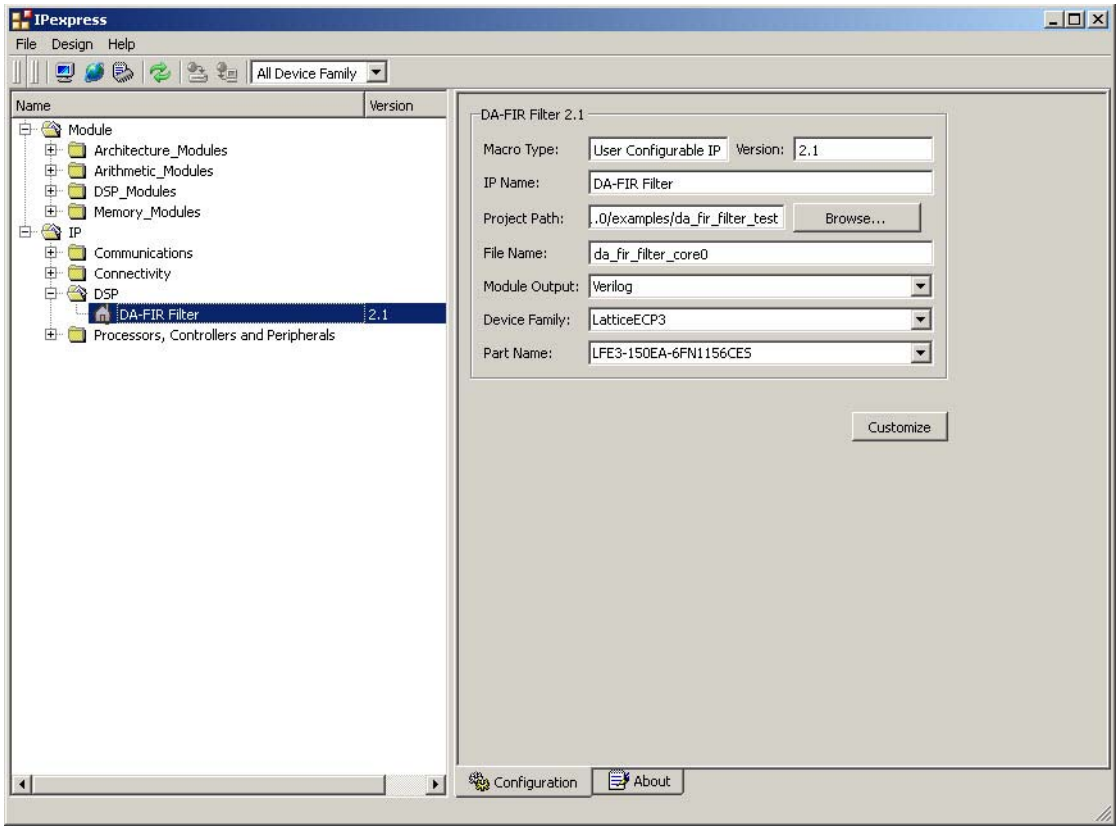
Getting Started

The DA-FIR Filter IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The ispLEVER IPexpress tool GUI dialog box for the DA-FIR Filter IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be loaded.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeSCM, Lattice ECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

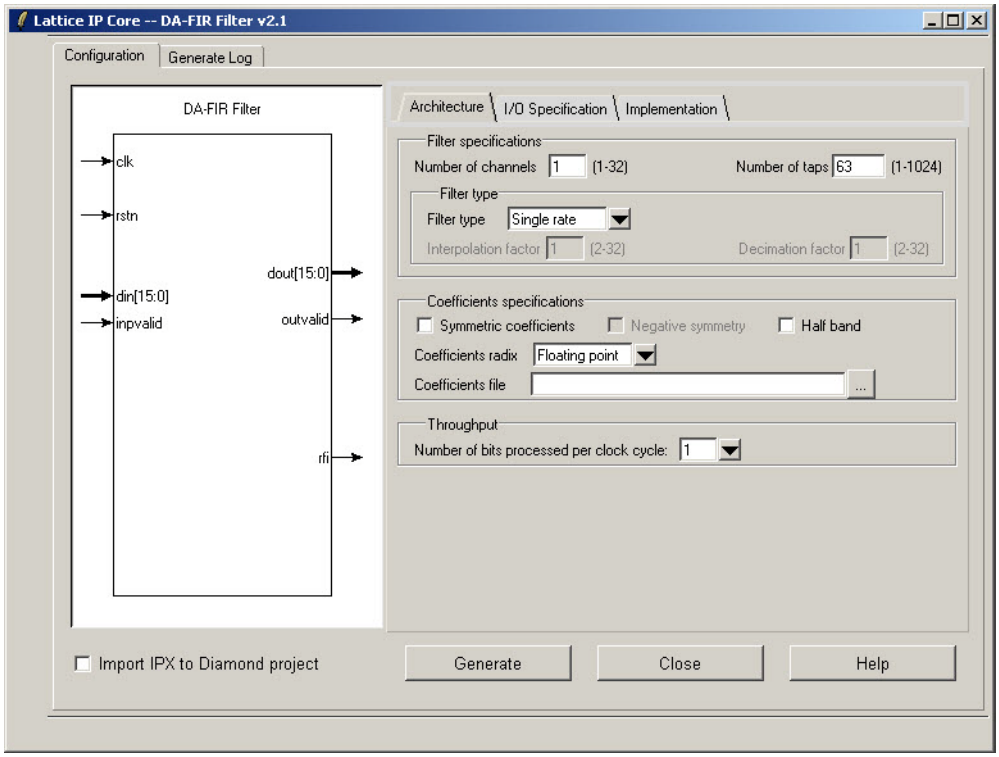
Figure 4-1. IPexpress Dialog Box (Diamond Version)



Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the DA-FIR Filter IP Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 15 for more information on the DA-FIR Filter IP parameter settings.

Figure 4-2. Configuration Dialog Box (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified “Project Path” directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. LatticeECP3 DA-FIR Filter IP Core Directory Structure

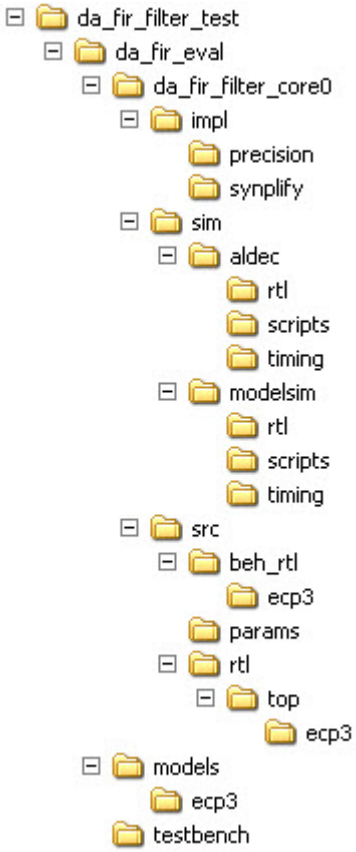


Table 4-1 provides a list of key files created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Description
<*username*>_inst.v	This file provides an instance template for the IP.
<*username*>_beh.v	This file provides a behavioral simulation model for the FFT core.
<*username*>_bb.v	This file provides the synthesis black box for the user's synthesis.
<*username*>.ngo	The ngo files provide the synthesized IP core.
<*username*>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<*username*>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
<*usernamee*>_top.[v,vhd]	This file provides a module which instantiates the FFT core. This file can be easily modified for the user's instance of the FFT core. This file is located in the `da_fir_eval/<username>/src/rtl/top` directory.
<*username*>_generate.tcl	This file is created when GUI "Generate" button is pushed and generation is invoked. This file may be run from command line.
<*username*>_generate.log	This is the IPexpress scripts log file.
<*username*>_gen.log	This is the IPexpress IP generation log file.

These are all of the files needed by the user to implement and verify the DA-FIR Filter IP core in their top-level design.

The \da_fir_eval and subtending directories provide files supporting DA-FIR core evaluation. The \da_fir_eval directory shown in Figure 4-3 contains files/folders with content that is constant for all configurations of the DA-FIR. The \<*username*> subfolder (\da_fir_filter_test in this example) contains files/folders with content specific to the username configuration.

The \da_fir_eval directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated. A \<*username*> directory is created by IPexpress each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<*username*> directory is generated for cores with different names, e.g. \da_fir_filer_core0, \da_fir_filer_core1, etc.

Instantiating the Core

The generated DA-FIR Filter IP core is provided in Lattice's proprietary .ngo format, which is independent of the HDL used to capture the rest of the user's design. The generated DA-FIR Filter IP core package includes black-box (<*username*>_bb.v) and instance (<*username*>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file is provided in C:\<*project_dir*>\da_fir_eval\<*username*>\src. Customers may use this top-level reference as the starting template for the top-level for their complete design.

Running Functional Simulation

The DA-FIR generator produces a cycle-accurate Verilog behavioral model of the user-configured DA-FIR Filter IP core, <*username*>_beh.v. The generated <*username*>_inst.v file provides a template for instantiating the model in a Verilog netlist. An example is shown below.

```
//============================================================================
// Verilog module generated by IPExpress    04/13/2006    14:17:22
// Filename: sample1_inst.v
// Copyright(c) 2005 Lattice Semiconductor Corporation. All rights reserved.
//============================================================================

//---------------------------------------------------------------
// sample1 simulation instance template
//---------------------------------------------------------------

sample1 sample1_inst (
    .clk             (clk),
    .rstn           (rstn),
    .d_in      (d_in[4:0]),
    .i_val         (i_val),
    .rdy            (rdy),
    .d_out   (d_out[11:0]),
    .o_val          (o_val)
    );
```

A few things to note:

- <*username*>_beh.v is generated for the specified configuration only. All the parameter settings have been resolved.
- The port connection list uses the port name as the default for the connection name, and the size of the connection is provided. In practice, users will edit these to connect to the wires in their netlists.
- Only the ports that are valid for this specific configuration are included in the port list (all possible ports are listed in the example).

Simulation Evaluation

The /da_fir_eval directory produced by the DA-FIR generator contains subdirectories for each requested configuration of the DA-FIR core. The contents of a simulation evaluation modelsim directory include:

- <*username*>_rtl_se.do – ModelSim execution script for functional simulation
- <*username*>_sdf_prec_se.do – ModelSim execution script for timing simulation
- <*username*>_sdf_synp_se.do – ModelSim execution script for timing simulation
- <*username*>_rtl.do – Aldec execution script for functional simulatoin
- <*username*>_sdf_prec.do – Aldec execution script for timing simulation
- <*username*>_sdf_synp.do – Aldec execution script for timing simulation
- stimulus_<*username*>.txt – The stimulus file for the simulation
- golden_<*username*>.txt – The golden file for the simulation.

Simulation environments are provided for both Mentor Graphics ModelSim and Aldec Active-HDL simulators.

Users may run the ModelSim evaluation simulation by doing the following:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose folder \<*project_dir*>\da_fir_eval\<*username*>\sim\modelsim\rtl.
3. Under the Tools tab, select **Execute Macro** and execute ..\scripts\<username>_rtl_se.do script.

Users may run the Active-HDL evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the Tools tab, select **Execute Macro**.

3.Browse to the directory \<*project_dir*>\da_fir_eval\<*username*>\sim\aldec\scripts and execute <*username*>_rtl.do script.

Synthesizing and Implementing the Core in a Top-Level Design

As mentioned previously, the DA-FIR Filter IP core itself is synthesized and is provided in NGO format when the core is generated. Users may synthesize the core in their own top-level design by instantiating the core in their top-level and then synthesizing the entire design with either Synplify or Precision RTL Synthesis.

The following steps are used in the implementation phase of the design process:

- Copy and paste the instance template (module_name_inst.v) into the user netlist where the DA-FIR Filter IP core resides (not limited to user's top level).
- Edit the connection list as necessary to connect the DA-FIR Filter IP core to the rest of the user design. (TIP: If the wire names in the user design match the port names of the DA-FIR Filter IP core, no editing is required. The file could be included in the design as is.)
- Run the synthesis tool, making sure the component definition file (module_name_bb.v) is included in the list of files to be compiled. The resulting netlist will contain a black box instantiation of the DA-FIR Filter IP core.
- When running map, place, and route, make sure the module_name.ngo and pmi_dp_ram*.ngo files can be found by the Diamond or ispLEVER software. This is accomplished either by copying the module_name.ngo file to the place-and-route working directory, or pointing to the directory where it resides. Set the directory where the module_name.ngo file resides by using one of the following methods:

– In Diamond, set the Macro Search Path property under **Project > Active Strategy > Translate Design Settings**.
– In ispLEVER, set the Macro Search Path property under **Process > Properties.**

If multiple identical IP cores are required in a single design, simply instantiate the same module multiple times with different connections. If multiple dissimilar IP cores are required, generate a different IP core for each type required using different module names, then instantiate each module type as required.

Things to consider:

- No timing model exists for the DA-FIR Filter IP core, so the synthesis tool will be unable to analyze the paths in and out of the core. The DA-FIR Filter IP core generator provides registers on all its outputs and assumes that its inputs are driven by registers clocked with the DA-FIR Filter IP core clock (clk).
- The DA-FIR Filter IP core netlist is not “lint-free” for all configurations, so warnings may be generated by the synthesis and map-place-route tools.

Implementation Evaluation

The DA-FIR Filter Generator optionally creates an implementation evaluation directory `<project_directory>/da_fir_eval/<username>/impl/synplify.`

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to `\<project_dir>\da_fir_eval\<username>\impl\synplify (or precision)` in the Open Project dialog box.
3. Select and open <*username*>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the **Process** tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project.**
2. Browse to `\<project_dir>\da_fir_eval\<username>\impl\synplify (or precision)` in the Open Project dialog box.
3. Select and open <*username*>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level entry in the left-hand GUI window.
5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The DA-FIR Filter IP core supports supports Lattice’s IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the **Build Database** process and choose **Properties** from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the T**arget** box.
4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate.** The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the Generate Log tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.

2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.
4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

Chapter 5:

Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

There are a number of ways to receive technical support.

Online Forums

The first place to look is Lattice Forums (http://www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA & Canada: 1-800-LATTICE (528-8423)
- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

E-mail Support

- techsupport@latticesemi.com
- techsupport-asia@latticesemi.com

Local Support

Contact your nearest Lattice Sales Office.

Internet

www.latticesemi.com

References

LatticeECP/EC

- HB1000, *LatticeECP/EC Family Handbook*

LatticeECP2/M

- HB1003, *LatticeECP2M Family Handbook*

LatticeECP3

- HB1009, *LatticeECP3 Family Handbook*

LatticeXP2

- HB1004, *LatticeXP2 Family Handbook*

LatticeSC/M

- DS1004, *LatticeSC/M Family Data Sheet*
- DS1005, *LatticeSC/M Family flexiPCS Data Sheet*

Revision History

Date	Document Version	IP Core Version	Change Summary
August 2006	01.0	1.0	Initial release.
February 2007	01.1	1.2	Added LatticeECP2 and LatticeECP2M™ appendices.
March 2007	01.2	1.3	Updated Features section.
			Updated default value for PERFORMANCE parameter in Parameter Descriptions table.
			Updated Hardware Evaluation section.
July 2007	01.3	1.4	Updated to support LatticeXP2 devices with ispLEVER 7.0.
April 2009	01.4	2.0	Updated to support DA-FIR Filter IP core version 2.0, including support for LatticeECP3 devices.
June 2010	01.5	2.0	Divided document into chapters. Added table of contents.
			Added Quick Facts table in Chapter 1, “Introduction.”
August 2010	1.6	2.1	Added Diamond software support throughout.

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the DA-FIR Filter IP core.

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP/EC Devices

Table A-1. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	296	340	0	481	192
1	9	Disable	8	TRUN	521	594	0	887	180
1	36	Enable	12	TRUN	590	689	0	899	174

1. Performance and utilization data are generated targeting an LFECP15E-4F256C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeECP/EC devices is DAFIR-GEN-E2-U2.

LatticeECP2 and LatticeECP2S Devices

Table A-2. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	317	378	0	481	341
1	9	Disable	8	TRUN	550	655	0	887	321
1	36	Enable	12	TRUN	625	743	0	899	320

1. Performance and utilization data are generated targeting an LFE2-20E-6F256C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2 family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeECP2/S devices is DAFIR-GEN-P2-U2.

LatticeECP2M and LatticeECP2MS Devices

Table A-3. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	317	378	0	481	343
1	9	Disable	8	TRUN	550	655	0	887	310
1	36	Enable	12	TRUN	625	743	0	899	291

1. Performance and utilization data are generated targeting an LFE2M20E-6F256C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeECP2M/S devices is DAFIR-GEN-PM-U2.

LatticeECP3 Devices

Table A-4. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	290	348	0	476	318
1	9	Disable	8	TRUN	512	611	0	877	279
1	36	Enable	12	TRUN	589	709	0	883	308

1. Performance and utilization data are generated targeting an LFE3-70E-7FN484CES device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeECP3 devices is DAFIR-GEN-E3-U2.

LatticeSC and LatticeSCM Devices

Table A-5. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	278	343	0	481	372
1	9	Disable	8	TRUN	564	759	0	895	338
1	36	Enable	12	TRUN	568	668	0	934	390

1. Performance and utilization data are generated targeting an LFSC3GA15E-6F256C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeSC/M family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeSC/M devices is DAFIR-GEN-SC-U2.

LatticeXP Devices

Table A-6. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	296	340	0	481	186
1	9	Disable	8	TRUN	521	594	0	887	178
1	36	Enable	12	TRUN	590	689	0	899	168

1. Performance and utilization data are generated targeting an LFXP10E-4F256C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeXP devices is DAFIR-GEN-XM-U2.

LatticeXP2 Devices

Table A-7. Performance and Resource Utilization[1]

Channels	Taps	Interpolation	Data Width	Rounding	Slices	LUTs	EBRs	Registers	f_{MAX} (MHz)
1	16	Disable	16	TRUN	317	378	0	481	274
1	9	Disable	8	TRUN	550	655	0	887	251
1	36	Enable	12	TRUN	625	743	0	899	270

1. Performance and utilization data are generated targeting an LFXP2-17E-6QN208C device using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.

Ordering Part Number

The Ordering Part Number (OPN) for the DA-FIR Filter Generator IP core targeting LatticeXP2 devices is DAFIR-GEN-X2-U2.