

Key Design Features

- Synthesizable, Technology independent VHDL Core
- Implemented as a systolic array for speed and scalability
- 50% less multipliers than a direct-form FIR implementation
- Support for filters with inverted symmetry such as High-pass, Differentiators and Hilbert Transformers
- Configurable coefficients, data width and number of taps
- Symmetric arithmetic rounding limits DC-bias problems
- Output saturation or wrap modes
- Supports 350 MHz+ sample rates¹

Applications

- High-speed filter applications where resources are limited
- General purpose FIR filters with symmetrical coefficients and an odd number of taps
- Filters that exhibit inverse symmetry in their coefficients such as High-pass, Differentiators and Hilbert transformers

Generic Parameters

Generic name	Description	Type	Valid range
num_taps	Number of filter taps	integer	> 2
dw	Width of input/output data samples	integer	≥ 2
cw	Width of coefficients	integer	≥ 2
fw	Number of coefficient fraction bits	integer	≥ 0 (fw < cw)
coeff [num_taps-1:0]	Filter coefficients	integer array	$-2^{32} < \text{coeff} < 2^{32}$
USE_ROUNDING	Use symmetric arithmetic rounding (not truncate)	Boolean	TRUE/FALSE
USE_SATURATE	Saturate outputs (not wrap)	Boolean	TRUE/FALSE
USE_INV_SYMMETRY	Coefficients have symmetry but inverted	Boolean	TRUE/FALSE
USE_OPTZERO	Optimize for zero-valued coefficients	Boolean	TRUE/FALSE

Block Diagram

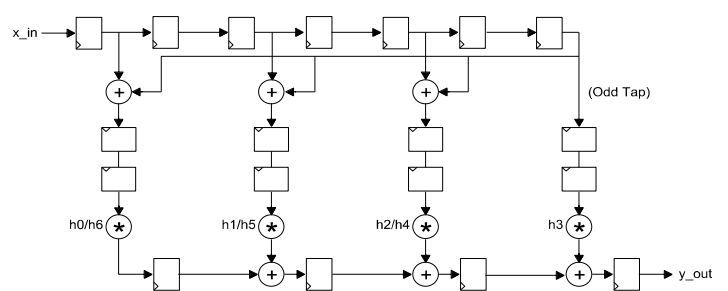


Figure 1: Symmetrical FIR filter architecture (Simplified)

Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Sampling clock	rising edge
en	in	Clock enable	high
x_in [dw-1:0]	in	Filter input samples (signed number)	data
y_out [dw-1:0]	out	Filter output samples (signed number)	data

General Description

FIR_NTAP_OSYM is an FIR filter with symmetrical coefficients and an odd number of taps. The architecture exploits the symmetry of the coefficients using half the number of multipliers compared to a normal FIR implementation. This allows for reduced area footprint while still maintaining the capacity for high sample-rates.

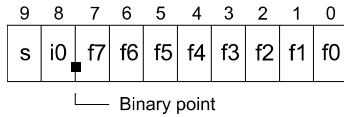
Organized as a systolic array (Figure 1) the filter is modular and fully scalable. Mathematically, the filter implements the difference equation:

$$y[n] = h_0 x[n] + h_1 x[n-1] + \dots + h_N x[n-N]$$

In the above equation, the input signal is $x[n]$, the output signal is $y[n]$ and h_0 to h_N represent the filter coefficients. The number N is the filter order, the number of filter taps being equal to $N + 1$. As the filter is symmetrical, the coefficient h_0 is equal to h_N , h_1 is equal to h_{N-1} , h_2 is equal to h_{N-2} etc. The filter has one unpaired tap having coefficient $h_{N/2}$.

¹ Xilinx Virtex5 FPGA used as a benchmark

Filter coefficients² are defined as signed fixed-point numbers in $[cw\ fw]$ format where cw is the total number of coefficient bits and fw is the number of bits in the fractional part. In all cases, cw must be at least 2 bits and fw must be less than cw to accommodate the sign bit. For instance, a coefficient in $[10\ 8]$ format would be arranged as follows:



The number of bits in the input and output samples is controlled by the parameter dw . Inputs and outputs are signed values (their format is purely relative). Output samples may be truncated to dw bits or rounded depending on the implementation option *USE_ROUNDING*. If the rounding option is selected, then symmetric arithmetic rounding is used. This means that the fraction 0.1000... is added to positive numbers and 0.0111... is added to negative numbers. Note that filters implemented with the rounding option will help to reduce the small amplitude offset introduced at DC (0 Hz baseband frequency) attributable to rounding error.

In addition, the option *USE_SATURATE* determines what will happen if the output samples are too large. If the saturate option is enabled, then in the event of an overflow, the output samples will saturate to the largest positive or negative number permitted by dw . With the saturate option disabled, the output samples will simply wrap around. Note that depending on the format of the coefficients and the data width relative to the magnitude of the input samples, the filter outputs may not overflow. In this case, the user may not require the saturation logic.

In the case where the filter coefficients are symmetrical but inverted about the center tap (i.e. $h_0 = -h_N$, $h_1 = -h_{N-1}$ etc.) the parameter *USE_INV_SYMMETRY* must be set. This ensures that the paired taps are subtracted as opposed to summed. Filters such as High-pass, Differentiators and Hilbert transformers exhibit this property.

Finally, the parameter *USE_OPTZERO* determines whether the design should be optimized for coefficients with a value of zero. This will result in a smaller design overall, but may also limit the best attainable clock speed.

Values are sampled on the rising clock-edge of clk when en is high. The latency of the filter is determined by the formula:

$$Lat_{TOT} = (Taps - 1)/2 + Lat_{RND} + Lat_{SAT} + 6$$

Where Lat_{TOT} is the total latency between the first input sample to enter the filter and the first output sample, $Taps$ is the total number of filter taps, Lat_{RND} is the latency of the rounding block (equal to 2 clock cycles) and Lat_{SAT} is the latency of the saturation block (equal to 1 cycle).

As an example, consider a 45 tap filter with rounding and saturation enabled. The total latency would be : $(45-1)/2 + 2 + 1 + 6 = 31$ clock cycles.

Functional Timing

Figure 2 shows a sequence of input samples for a 45 tap filter with rounding and saturation enabled. Output samples appear 31 clock-cycles later.

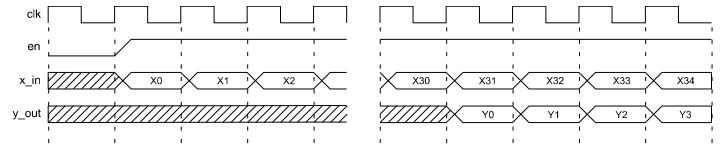


Figure 2: FIR filter input/output samples

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file. Note that all generic parameters are defined in the package 'fir_ntap_osym_pack.vhd'.

Source file	Description
fir_ntap_osym_pack.vhd	Package containing all generic parameters - including coefficients.
fir_ntap_osym_mad.vhd	Multiply-Add block
fir_ntap_osym_mad_zero.vhd	Multiply-Add block optimized for zero-valued coefficients
fir_ntap_osym_madl.vhd	Multiply-Add block (Odd tap)
fir_ntap_osym_madl_zero.vhd	Multiply-Add block (Odd tap) optimized for zero-valued coefficients
fir_ntap_osym_rnd.vhd	Rounding block
fir_ntap_osym_sat.vhd	Saturation block
fir_ntap_osym.vhd	Top-level block
fir_ntap_osym_bench.vhd	Top-level test bench

Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. fir_ntap_osym_pack.vhd
2. fir_ntap_osym_mad.vhd
3. fir_ntap_osym_mad_zero.vhd
4. fir_ntap_osym_madl.vhd
5. fir_ntap_osym_madl_zero.vhd
6. fir_ntap_osym_rnd.vhd
7. fir_ntap_osym_sat.vhd
8. fir_ntap_osym.vhd
9. fir_ntap_osym_bench.vhd

The VHDL testbench instantiates the FIR filter component. The user may modify the generic parameters in the file 'fir_ntap_osym_pack.vhd' as required. The test provided is configured for a Hilbert Transform filter with 45 taps.

² The design is supplied with Matlab® scripts for the easy generation of coefficient sets using FDAtool®. Please see application note: app_note_zc002.pdf for more details.

The simulation must be run for at least 1 ms during which time the impulse response and step response of the filter is tested. The simulation generates a text file called 'fir_ntap_osym_out.txt' that contains the output samples captured during the course of the test.

Figures 3 and 4 respectively contain the impulse response and step response outputs of the filter.

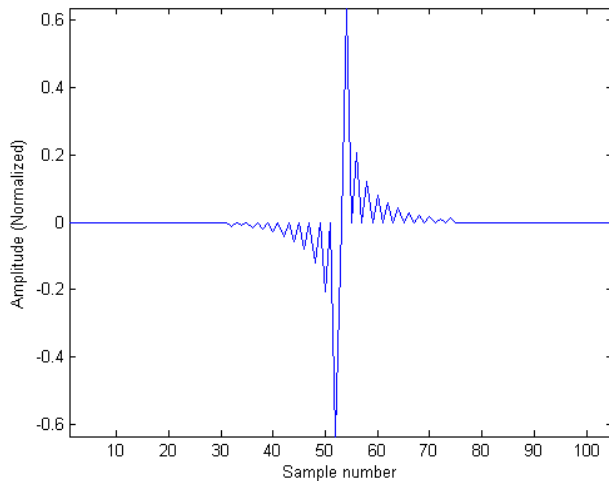


Figure 3: Impulse response of the 45-tap Hilbert Transform filter

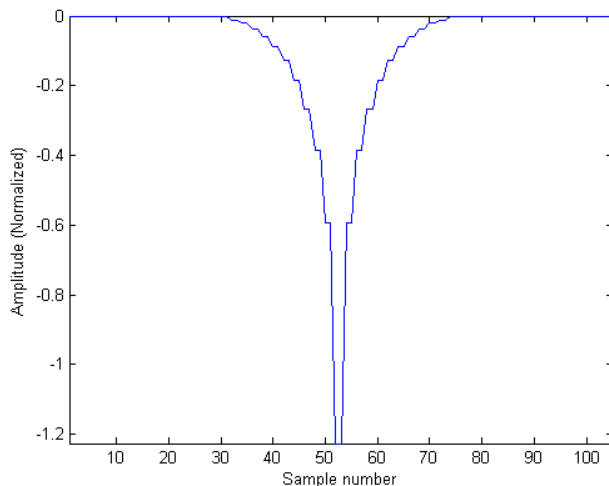


Figure 4: Step response of the 45-tap Hilbert Transform filter

Synthesis

The files required for synthesis and the design hierarchy is shown below:

- fir_ntap_osym_pack.vhd
- fir_ntap_osym.vhd
 - fir_ntap_osym_mad.vhd
 - fir_ntap_osym_mad_zero.vhd
 - fir_ntap_osym_madl.vhd
 - fir_ntap_osym_madl_zero.vhd
 - fir_ntap_osym_rnd.vhd
 - fir_ntap_osym_sat.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison.

Trial synthesis results are shown with the generic parameters set to: num_taps = 45, dw = 16, cw = 10, fw = 9, USE_ROUNDING = TRUE, USE_SATURATION = TRUE, USE_INV_SYMMETRY = TRUE, USE_OPTZERO = TRUE. Resource usage is specified after Place and Route.

VIRTEX 5

Resource type	Quantity used
Slice register	254
Slice LUT	447
Block RAM	0
DSP48	11
Clock frequency (worst case)	287 MHz
Clock frequency (best case)	365 MHz

STRATIX III

Resource type	Quantity used
Register	1278
ALUT	160
Block Memory bit	0
DSP block 18	22
Clock frequency (worse case)	287 MHz
Clock frequency (best case)	370 MHz

Revision History

Revision	Change description	Date
1.0	Initial revision	19/05/2008
1.1	Fixed document typos	10/07/10
1.2	Added description fro the generic parameter USE_OPTZERO. Updated synthesis results	09/02/2011