Key Design Features

- Synthesizable, technology independent VHDL Core
- Implemented as a systolic array for speed and scalability
- Configurable coefficients, data width and number of taps
- Symmetric arithmetic rounding limits DC-bias problems
- Output saturation or wrap modes
- Supports 550 MHz+ sample rates[1]

Applications

- Very high-speed filtering applications
- General purpose FIR filters with odd or even numbers of taps
- Filters with arbitrary sets of coefficients (e.g. non-symmetrical)

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

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	≥ 0
USE_ROUNDING	Use symmetric arithmetic rounding (not truncate)	Boolean	TRUE/FALSE
USE_SATURATE	Saturate outputs (not wrap)	Boolean	TRUE/FALSE
USE_OPTZERO	Optimize for zero-valued coefficients	Boolean	TRUE/FALSE

1 Xilinx Virtex 5 FPGA used as a benchmark

Block Diagram

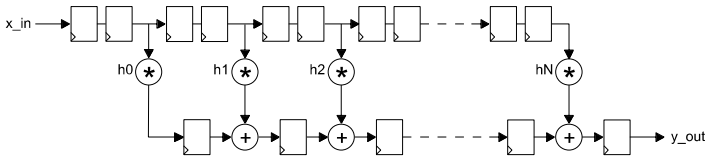


Figure 1: FIR filter architecture (Simplified)

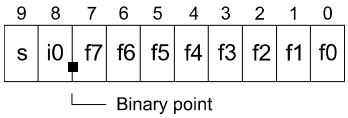
General Description

FIR_NTAP is an FIR filter implementation designed for very high sample rate applications. Organized as a systolic array (Figure 1) the filter is modular and fully scalable, permitting the user to specify large order filters without compromising maximum attainable clock-speed. Mathematically, the filter implements the difference equation:

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

In the above equation, the input signal is *x*[*n*], the output signal is *y*[*n*] and *h0* to *hN* represent the filter coefficients. The number *N* is the filter order, the number of filter taps being equal to *N*+1.

Filter coefficients[2] 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.

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.

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:

$$Lat_{TOT} = Taps + Lat_{RND} + Lat_{SAT} + 4$$

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 20 tap filter with rounding and saturation enabled. The total latency would be : 20 + 2 + 1 + 4 = 27 clock cycles.

Functional Timing

Figure 2 shows a sequence of input samples for a 20 tap filter with rounding and saturation enabled. Output samples appear 27 clock-cycles later.

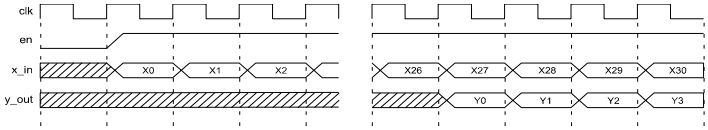


Figure 2: FIR filter input/output samples

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_pack.vhd'.

Source file	***Description***
fir_ntap_pack.vhd	Package containing all generic parameters - including coefficients.
fir_ntap_mad.vhd	Multiply-Add block
fir_ntap_mad_zero.vhd	Miltiply-Add block optimized for zero-valued coefficients
fir_ntap_rnd.vhd	Rounding block
fir_ntap_sat_vhd	Saturation block
fir_ntap.vhd	Top-level block
fir_ntap_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_pack.vhd
2. fir_ntap_mad.vhd
3. fir_ntap_mad_zero.vhd
4. fir_ntap_rnd.vhd
5. fir_ntap_sat.vhd
6. fir_ntap.vhd
7. fir_ntap_bench.vhd

The VHDL testbench instantiates the FIR filter component. The user may modify the generic parameters in the file 'fir_ntap_pack.vhd' as required. The test provided is configured for an low-pass equiripple filter with 45 taps.

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_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.

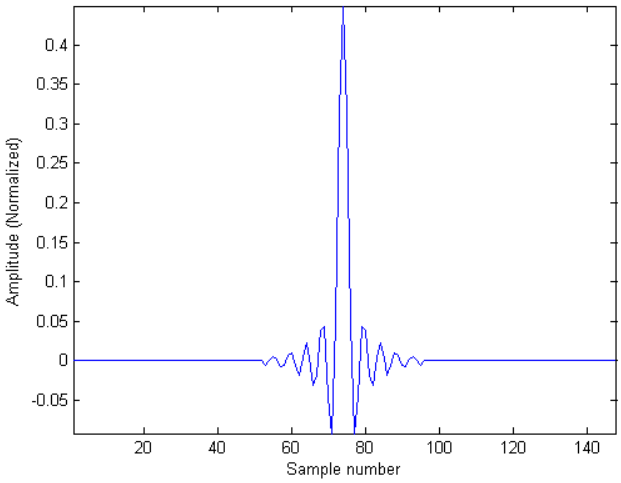


Figure 3: Impulse response of the 45-tap low-pass FIR filter