

### Key Design Features

- Synthesizable, technology independent VHDL Core
- N-channel FIR filter core implemented as a systolic array for speed and scalability
- Support for one or more independent channels (8 maximum)
- Configurable coefficients, data width and number of taps
- Symmetric arithmetic rounding limits DC-bias problems
- Output saturation or wrap modes
- Much cheaper than implementing separate FIR filters in parallel as hardware resources are shared between all channels
- Supports 500 MHz+ sample rates (500/N MHz per channel)<sup>1</sup>

### Applications

- High-speed filtering applications where hardware resources are limited - e.g. when it becomes impractical to use multiple FIR filters in parallel
- Dual-channel inputs such as complex valued I/Q in digital communications systems
- Parallel DSP processor architectures
- General purpose FIR filters with odd or even numbers of taps
- Filters with arbitrary sets of coefficients (e.g. non-symmetrical)

### Generic Parameters

Generic name	Description	Type	Valid range
num_channels	Number of filter channels	integer	$1 \leq N \leq 8$
num_taps	Number of filter taps	integer	$> 2$
dw	Width of input/output data samples	integer	$\geq 2$
cw	Width of coefficients	integer	$\geq 2$
fw	Number of coefficient fraction bits	integer	$\geq 0$ ( $fw < cw$ )
coeff [num_taps-1:0]	Filter coefficients	integer array	$\geq 0$
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

### Block Diagram

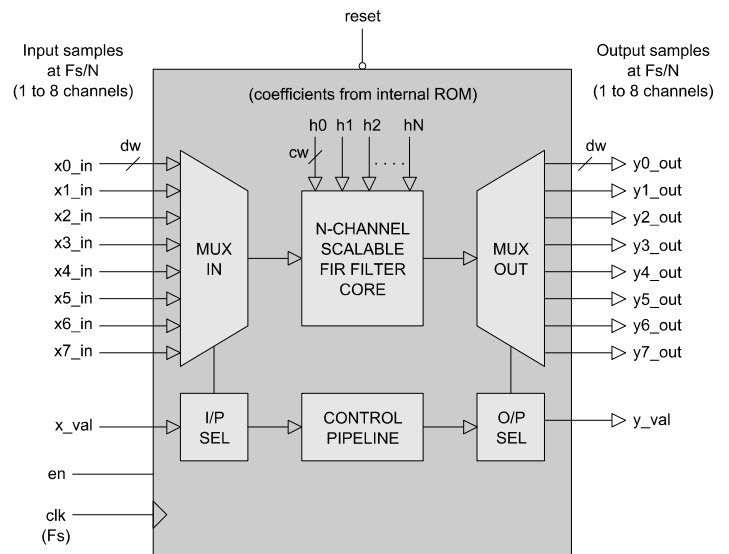


Figure 1: N-Channel FIR Filter Architecture

### Pin-out Description

Pin name	I/O	Description	Active state
clk	in	System clock ( $F_s$ )	rising edge
reset	in	System reset	low
en	in	Clock enable	high
x_val	in	Filter inputs valid (coincident with first valid input sample at x0_in)	high
xN_in [dw - 1:0]	in	Filter input samples (signed number)	data
y_val	out	Filter outputs valid (coincident with first valid output sample at y0_out)	high
yN_out [dw - 1:0]	out	Filter output samples (signed number)	data

### General Description

FIR\_NTAP\_MUX is an N-channel multiplexed FIR filter designed for high sample rate applications where hardware resources are limited. The main filter core is organized as a scalable systolic array permitting the user to specify large order filters without compromising maximum attainable clock-speed.

Essentially the filter functions as if it were 'N' separate FIR filters in parallel. Each input sample being multiplexed into the filter at a sample rate equal to  $F_s/N$ , where  $F_s$  is the sampling frequency of the main filter core. Likewise, output samples are updated at a frequency of  $F_s/N$ .

The first sample into the filter is aligned by asserting the signal *x\_val* high. The signal *y\_val* is asserted with the first valid output sample.

<sup>1</sup> Xilinx Virtex 5 FPGA used as a benchmark

Data samples are advanced in the pipeline on the rising clock-edge of *clk* when *en* is active high. When *en* is low then all data samples are stalled. The clock-enable signal may be used to temporarily disable the filter - or possibly to modify the effective sampling frequency of the system clock. If the clock-enable is not needed it is recommended that this signal be tied high as it will improve overall circuit performance.

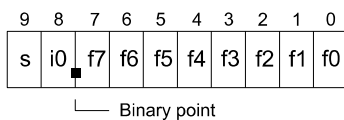
Mathematically, the filter implements the difference equation:

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

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  $N+1$ .

### Filter coefficients and I/O specification

Filter coefficients<sup>2</sup> 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). Unused inputs should be tied to zero. For instance, if a filter design only requires four channels, then inputs  $x4\_in$  to  $x7\_in$  should be tied low.

### Filter implementation options

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.

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.

<sup>2</sup> 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.

### Filter latency

The latency of the filter defined here is the latency in system clock-cycles from the point in which the first input sample is valid, to the point in which the first output sample is valid. The total latency is defined by the following formula:

$$Lat_{TOT} = (N * Taps) + (N * 2) + Lat_{RND} + Lat_{SAT} + 2$$

$N$  = Number of channels

$Taps$  = Number of filter taps

$Lat_{RND} = 2$  if rounding enabled, 0 otherwise

$Lat_{SAT} = 1$  if saturate enabled, 0 otherwise

As an example, consider a 4-channel, 50 tap filter with rounding and saturation enabled. The total latency would be calculated as:  $(4*50) + (4*2) + 2 + 1 + 2 = 213$  clock cycles.

### Sampling frequency considerations

The system clock frequency is the sampling frequency of the internal filter core. Let this be denoted as  $F_S$ . It follows that the sampling frequency of the input and output samples is dependent on the number of multiplexed channels,  $N$ . In particular the following formula must be observed for correct filter operation:

$$F_S(\text{one channel}) = \frac{F_S}{N}$$

### Functional Timing

Figure 2 shows a sequence of input samples for an 8-channel filter. Note that the signal *x\_val* is used to align the first data sample at the filter input. From that point onwards, the remaining inputs are sampled sequentially in turn. If the user wishes to re-align the filter inputs, then a system reset must be performed before *x\_val* is reasserted with the new first sample.

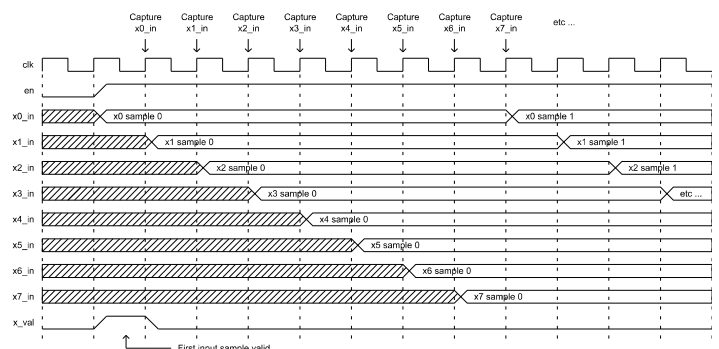


Figure 2: Input timing - all 8 channels active

The output samples follow a similar pattern. Figure 3 shows the corresponding outputs for an 8-channel filter. From the point at which *y\_val* is asserted, the downstream circuit must sample the filter outputs on consecutive clock cycles.

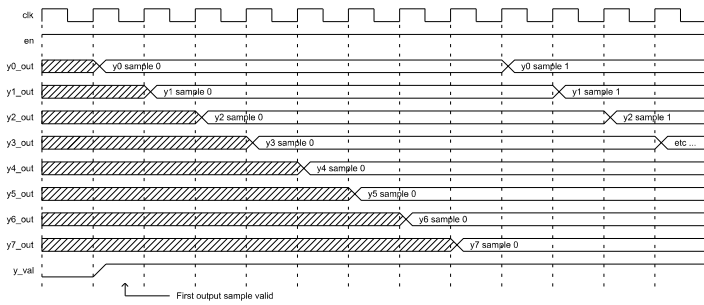


Figure 3: Output timing - all 8 channels active

### 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\_mux\_pack.vhd'.

Source file	Description
fir_ntap_mux_pack.vhd	Package containing all generic parameters - including coefficients
fir_ntap_mux_in.vhd	Input sample multiplexer
fir_ntap_mux_out.vhd	Output sample de-multiplexer
fir_ntap_mux_val_pipe.vhd	Control / valid pipeline
fir_ntap_mux_mad.vhd	Multiply-add block
fir_ntap_mux_mad_zero.vhd	Multiply-add block optimized for zero-valued coefficients
fir_ntap_mux_reg.vhd	Register delay element
fir_ntap_mux_rnd.vhd	Rounding block
fir_ntap_mux_sat.vhd	Saturation block
fir_ntap_mux.vhd	Top-level component
fir_ntap_mux_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\_mux\_pack.vhd
2. fir\_ntap\_mux\_in.vhd
3. fir\_ntap\_mux\_out.vhd
4. fir\_ntap\_mux\_val\_pipe.vhd
5. fir\_ntap\_mux\_mad.vhd
6. fir\_ntap\_mux\_mad\_zero.vhd
7. fir\_ntap\_mux\_reg.vhd
8. fir\_ntap\_mux\_rnd.vhd
9. fir\_ntap\_mux\_sat.vhd
10. fir\_ntap\_mux.vhd
11. fir\_ntap\_mux\_bench.vhd

The VHDL testbench instantiates the FIR filter component. The user may modify the generic parameters in the file 'fir\_ntap\_mux\_pack.vhd' as required.

The test provided is configured for an 8-channel band-pass filter design with 31 taps and a sampling frequency of 80MHz. The magnitude response of the filter is shown in Figure 4.

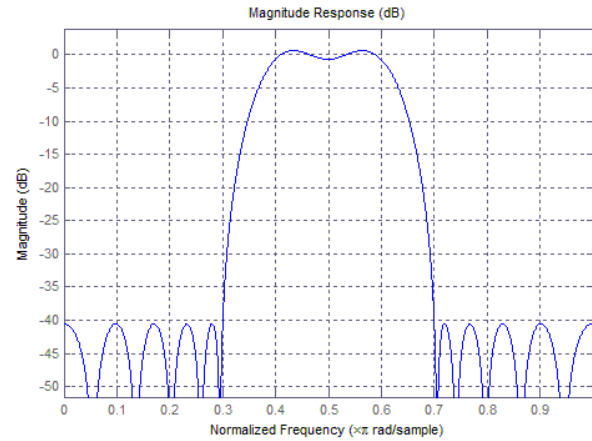


Figure 4: FIR band-pass filter magnitude response (same for all 8 channels)

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 series of 8 text files called 'fir0\_out.txt', 'fir1\_out.txt', 'fir2\_out.txt', etc .. These files contain the output samples for all 8 filter channels captured during the course of the test.

Figures 5 and 6 respectively demonstrate the impulse response and step response outputs for the given test example.

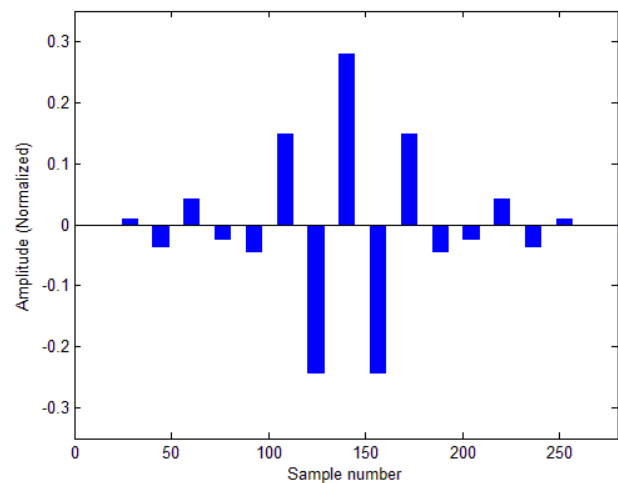


Figure 5: FIR band-pass filter impulse response (all 8 channels)

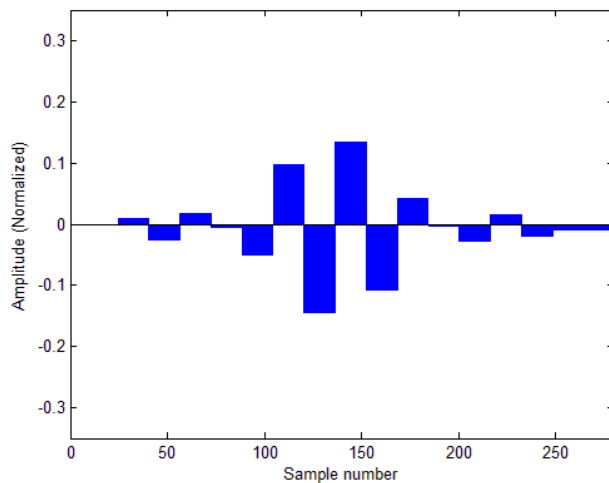


Figure 6: FIR band-pass filter step response  
(all 8 channels)

## Synthesis

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

- fir\_ntap\_mux\_pack.vhd
- fir\_ntap\_mux.vhd
  - fir\_ntap\_mux\_in.vhd
  - fir\_ntap\_mux\_val\_pipe.vhd
  - fir\_ntap\_mux\_mad.vhd
  - fir\_ntap\_mux\_mad\_zero.vhd
  - fir\_ntap\_mux\_reg.vhd
  - fir\_ntap\_mux\_rnd.vhd
  - fir\_ntap\_mux\_sat.vhd
  - fir\_ntap\_mux\_out.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.

Smaller and faster designs will be achieved by setting the parameters `USE_ROUNDING` and `USE_SATURATION` to `FALSE`. In addition, the filter source code is designed so that any coefficients with the value zero are optimized away during synthesis. This results in a design with fewer hardware multipliers. To enable this mode then the generic parameter `USE_OPTZERO` must be set to `TRUE`.

Fixing the clock-enable signal to logic '1' will generally result in a faster and more compact filter implementation. Unused data inputs should be tied to logic '0'.

Trial synthesis results are shown with the generic parameters set to: `num_channels = 8`, `num_taps = 31`, `dw = 16`, `cw = 10`, `fw = 9`, `USE_ROUNDING = FALSE`, `USE_SATURATION = FALSE`, and `USE_OPTZERO = TRUE`.

Resource usage is specified after Place and Route.

### VIRTEX 5

Resource type	Quantity used
Slice register	1051
Slice LUT	701
Block RAM	0
DSP48	15
Clock frequency (worst case)	440 MHz
Clock frequency (best case)	533 MHz

### STRATIX III

Resource type	Quantity used
Register	2597
ALUT	1364
Block Memory bit	11678
DSP block 18	0
Clock frequency (worse case)	280 MHz
Clock frequency (best case)	350 MHz

## Revision History

Revision	Change description	Date
1.0	Initial revision	08/02/2011
1.1	Added filter output plots	23/03/2011