Key Design Features

- Synthesizable, technology independent VHDL Core
- Fully pipelined architecture for highest possible sample rates
- 16-bit signed input and output samples
- 16-bit fixed-point coefficients and scaling factor
- Internal overflow detection
- SOS coefficient matrix maps directly to filter coefficients
- Cascade SOS blocks in series for higher-order filters
- Small implementation size - requiring only 6 H/W multipliers

Applications

- General purpose low-pass, band-pass and high-pass filters
- Specialist filters including: peaking, notching, all-pass, group delay equalization, arbitrary magnitude etc.
- IIR filtering in higher sample-rate applications
- High performance filtering when resources are limited

Pin-out Description

Pin name	***I/O***	***Description***	***Active state***
clk	in	Sampling clock	rising edge
reset	in	Asynchronous reset	low
en	in	Clock enable	high
coeff_s0 [15:0]	in	16-bit Scale factor in signed [16 13] format	data
coeff_b0 [15:0]	in	Coefficient b0 in signed [16 13] format	data
coeff_b1 [15:0]	in	Coefficient b1 in signed [16 13] format	data
coeff_b2 [15:0]	in	Coefficient b2 in signed [16 13] format	data
coeff_a1 [15:0]	in	Coefficient a1 in signed [16 13] format	data
coeff_a2 [15:0]	in	Coefficient a2 in signed [16 13] format	data
x_in [15:0]	in	Filter input samples as a 16-bit signed number	data
y_out [15:0]	out	Filter output samples as a 16-bit signed number	data
overflow	out	Output overflow flag	high

Block Diagram

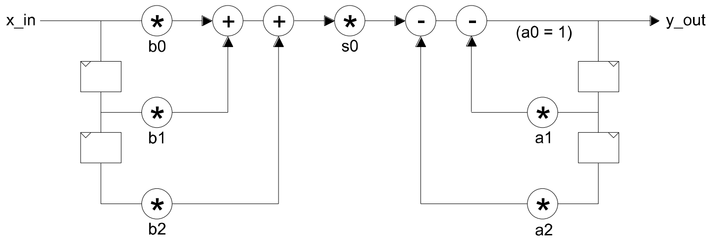


Figure 1: IIR filter SOS (Simplified)

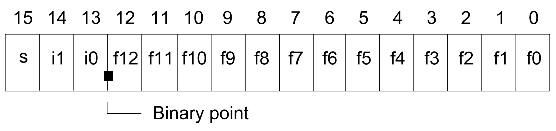
General Description

IIR_SOS is a 2nd order IIR filter sometimes referred to as a 'bi-quad'. Internally, it has a fully pipelined architecture permitting the highest possible sample rates for IIR filtering. The IIR block is modular and any number of SOS blocks may be joined in series to implement higher order filters. Mathematically, the filter implements the difference equation:

$$y[n] = s_0(b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]) \\ - a_1 y[n-1] - a_2 y[n-2]$$

In the above equation, the input signal is *x*[*n*], the output signal is *y*[*n*] and *b0*, *b1*, *b2*, *a1*, *a2* represent the filter coefficients. The value *s0* is a scale factor that is often required in stable IIR implementations to prevent the filter oscillating.

Filter coefficients are 16-bits wide and defined as signed fixed-point numbers in [*16 13*] format - i.e. 1 sign-bit, 2 integer bits and 13 faction bits. The scale factor is also specified in the same format. The diagram below shows this pictorially:



Input and output samples are signed 16-bit values (their format is purely relative). During normal operation of the filter, if the output samples are too large to be accommodated within a 16-bit signed number, then the *overflow* flag will be asserted and the output samples will saturate to the largest positive or negative value. These are respectively 32767 and -32768.

Generally, if an overflow occurs, then it indicates an unstable set of filter coefficients that have caused the filter to oscillate. Alternatively, small overflows may be prevented by adjusting the input gain to the filter.

Values are sampled on the rising clock-edge of *clk* when *en* is high. The latency of the IIR filter between the first input sample and the first output sample is 7 clock cycles. Note that it is recommended that before normal operation begins, the filter should be reset by asserting the *reset* signal low for at least one clock cycle. This ensures that the feedback paths of the filter are initialized to zero.

Functional Timing

Figure 2 shows a sequence of input samples for the IIR 2nd order section Output samples appear 7 clock cycles later.

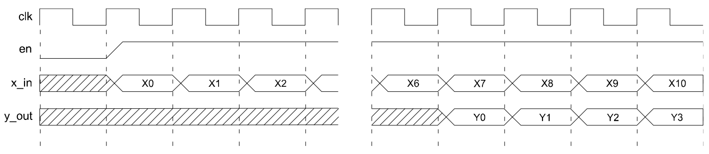


Figure 2: IIR SOS filter input/output samples

Figure 3 demonstrates an example of a positive overflow condition where the output sample saturates to the maximum positive number (0x7FFF) for a signed 16-bit value.

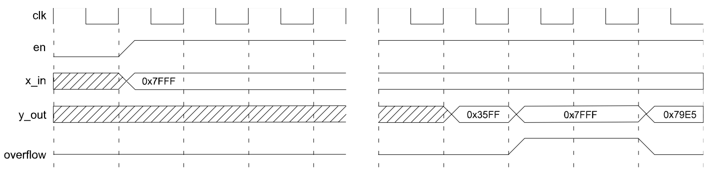


Figure 3: IIR filter positive overflow and output saturation

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	**Description**
iir_sos.vhd	Top-level block
iir_sos_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. iir_sos.vhd
2. iir_sos_bench.vhd

The VHDL testbench instantiates the IIR_SOS filter component. The user may modify the coefficients as required in order to implement the desired filter response. The test provided is configured for a 2nd order low-pass Butterworth filter with a sample rate of 100MHz. Coefficients for a high-pass Chebyshev, a band-pass Butterworth and an all-pass arbitrary group-delay filter are also provided as examples.

The simulation must be run for at least 100 us during which time the impulse response and step response of the filter is tested.

The simulation generates a text file called: 'iir_sos_out.txt'. This file contains the output samples captured during the course of the test. Figure 4 shows the magnitude, impulse and step response outputs of the low-pass Butterworth example.

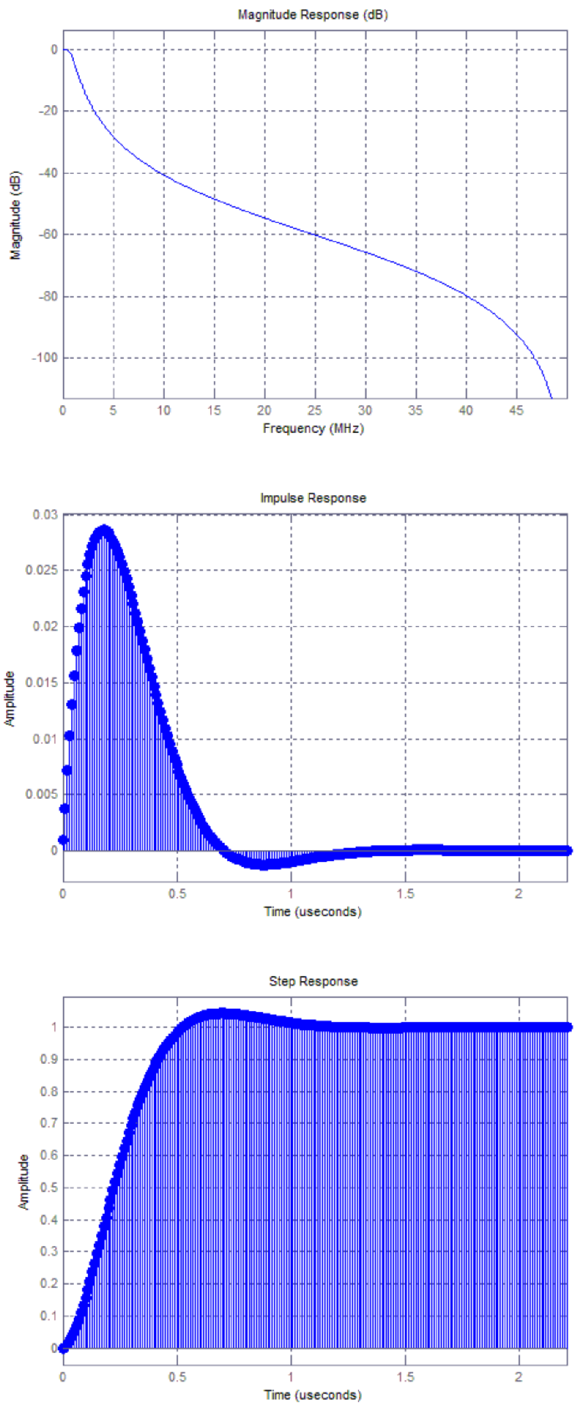


Figure 4: 2nd-order low-pass Butterworth filter responses

Synthesis

The source file 'iir_sos.vhd' is the only one required for synthesis. There are no sub-modules in the design.

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx® Virtex 6 and Spartan 6 FPGA devices. Synthesis results for other FPGAs and technologies can be provided on request.

Resource usage is specified after Place and Route.

VIRTEX 6

Resource type	***Quantity used***
Slice register	214
Slice LUT	179
Block RAM	0
DSP48	6
Occupied slices	70
Clock frequency (approx)	180 MHz

SPARTAN 6

Resource type	***Quantity used***
Slice register	219
Slice LUT	172
Block RAM	0
DSP48	6
Occupied slices	76
Clock frequency (approx)	120 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	20/05/2008
1.1	Increased coefficient width to 16-bits. Updated synthesis results in line with code changes	03/01/2012
1.2	Updated synthesis results for Xilinx® 6 series FPGAs	23/07/2012