

Key Design Features

- Synthesizable, technology independent VHDL Core
- 16-bit signed input and output data samples
- Accepts either complex or real inputs
- Precision Digital Down Converter with 110 dB SFDR
- DDS Frequency resolution of $F_s/2^{32}$ (F_s = sample frequency)
- DDS Phase resolution of $2\pi/2^{16}$
- Programmable tone centre frequency
- Programmable detection bandwidths
- Detection bandwidth range from 0.005 to 0.04 x F_s
- Choice of low pass filter responses
- Typical FPGA sample rates of up to 150 MHz¹

Applications

- Touch tone decoding (e.g. DTMF tones)
- Precision frequency monitoring and control
- Complex digital down conversion
- FSK/OOK/ASK demodulation

Generic Parameters

Generic name	Description	Type	Valid range
dithering	Enable phase dithering in DDS component	boolean	TRUE/FALSE
seed	Seed for random number generator in DDS component	std_logic vector	$0 < \text{seed} < 2^{32}$
use_complex	Enable Complex or Real data samples	boolean	TRUE: use ports i_{in} and q_{in} FALSE: use port i_{in} only
filter_type	Low-pass filter response type	integer	0: Narrow BW 1: Medium BW 2: Wide BW 3: V. Wide BW
threshold	Tone detect threshold	integer	0 to 65535

Block Diagram

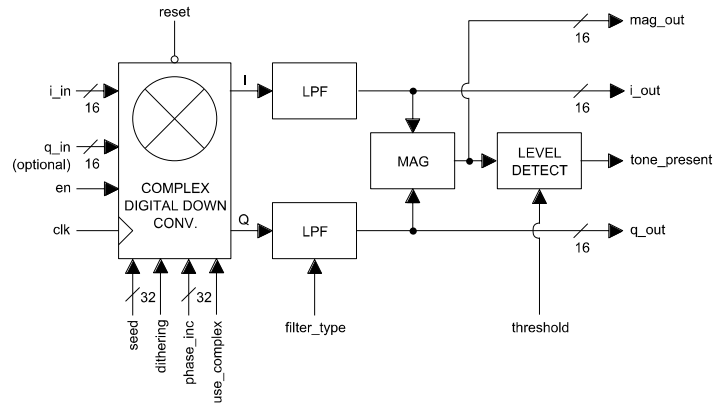


Figure 1: Tone decoder architecture

Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Sample clock	rising edge
reset	in	Asynchronous reset	low
en	in	clock enable	high
phase_inc [31:0]	in	Phase increment as an unsigned 32-bit number (controls tone centre frequency)	data
i_in [15:0]	in	Real (In-phase) input data samples as a 16-bit signed number	data
q_in [15:0]	in	Imaginary (Quadrature) input data samples as a 16-bit signed number	data
i_out [15:0]	out	Output data samples as a 16-bit signed number	data
q_out [15:0]	out	Output data samples as a 16-bit signed number	data
mag_out[15:0]	out	Magnitude of complex output	data
tone_present	out	Tone present flag	high

General Description

TONE_DEC is a precision Tone Decoder/Digital Down Converter with the capacity to support either real or complex data samples. Samples are first down-converted to baseband before subsequent filtering and tone detection. Figure 1 shows the basic architecture and signal paths.

The centre frequency of the tone is fully programmable and is generated by a local oscillator (DDS). The DDS has an SFDR of better than 100dBs (with phase dithering) and a theoretical SNR of approximately 100dBs².

¹ Altera Stratix III FPGA used as a benchmark

² See the dds.pdf datasheet on the ZIPcores website for more details

After down-conversion, the I and Q signal paths are filtered to remove components above the tone of interest. The characteristics of these filters may be changed depending on the desired detection bandwidth and response time.

Finally, a power function is used to compute the relative magnitude of the signal after filtering. If the complex input signal contains the tone of interest, then the relative output magnitude, *mag_out*, will be greater. The generic parameter *threshold*, is a 16-bit unsigned integer that generates the *tone_present* flag when the magnitude has reached a sufficient level.

Tone Centre Frequency

The frequency of the local oscillator is controlled by the signal *phase_inc*. The phase increment may be calculated using the formula:

$$\Phi_{inc} = (F_{out} \cdot 2^{32}) / F_s + 0.5$$

Where F_{out} is the desired waveform output frequency and F_s is the sampling frequency. Note that an *integer* value for the phase increment must be used. As an example, consider a 100 MHz sample clock with a desired local oscillator frequency of 6.197 MHz. The phase increment would be calculated as $(6.197 * 2^{32}) / 100 + 0.5 = 266159123$.

The minimum and maximum local oscillator frequencies are given by the following formulas. As an example, a 100 MHz sample clock would allow a minimum local oscillator frequency of 0.0233 Hz. Conversely, the maximum frequency the local oscillator can generate is given by the Nyquist/Shannon sampling theorem ($F_s/2$).

$$F_{min} = F_s / 2^{32} \quad , \quad F_{max} = F_s / 2$$

Low pass I/Q filters

After digital down conversion, the I and Q paths are filtered using a pair of IIR filters. Note that as these filters are recursive in nature, then the resulting output phase is non-linear. This must be taken into consideration if subsequent signal processing is to be done on the complex outputs.

In total, there are four separate filter responses that may be selected using the generic parameter *filter_type*. Figure 2 shows the different filter responses for each setting.

Filter (a) is characterized by a very narrow bandwidth and a long impulse response time. Conversely, filter (d) has a much wider bandwidth but a very short response time. The table below outlines these parameters in more detail.

Filter type	-3dB cutoff frequency	Approximate Response time
0 : (a)	$0.005 * (F_s / 2)$	300 samples
1: (b)	$0.01 * (F_s / 2)$	150 samples
2 : (c)	$0.02 * (F_s / 2)$	75 samples
3 : (d)	$0.04 * (F_s / 2)$	35 samples

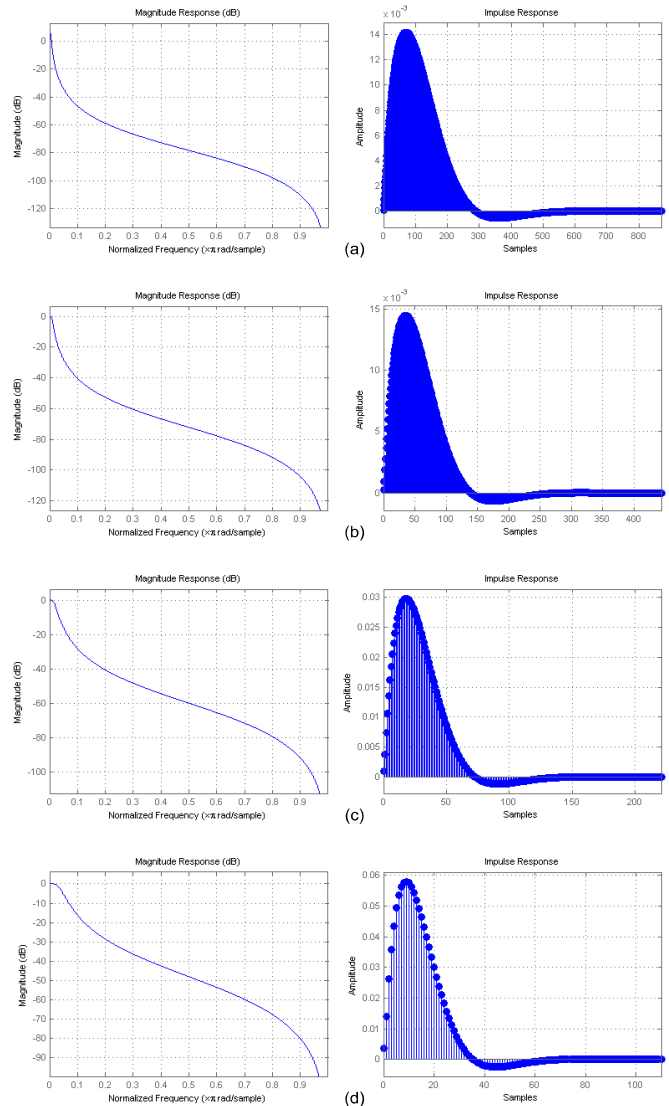


Figure 2: Low-pass filter with selectable characteristics. The -3dB cutoff points are: (a) 0.005, (b) 0.001, (c) 0.02 and (d) 0.04 rads/sample

Note that it is important that the full 16-bit dynamic range of the tone decoder inputs is used in order for the low-pass filters to function optimally. For input samples with lower numbers of significant bits, the data samples should be left-shifted so that the full 16-bit dynamic range is utilized.

Functional Timing

Figure 3 shows the operation of the tone decoder during normal operation. In this particular example, the threshold has been set to 0x400 and *use_complex* has been set to *false*. This means that only the 'I' signal path is used with 'Q' unused.

Notice that the *tone_present* flag is asserted high when the output magnitude exceeds the threshold. The threshold value may be adjusted in order to alter the sensitivity of the *tone_present* flag. This may be necessary as the filters have slightly different passband gain. The waveforms also show the action of the clock-enable signal *en*. Samples are clocked on a rising clock-edge when *en* is high.

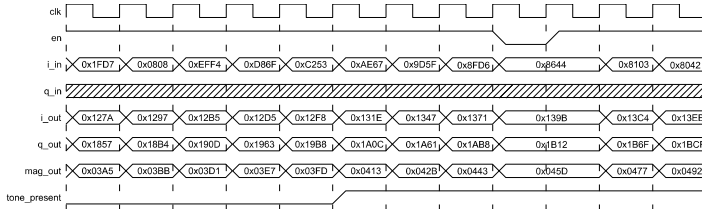


Figure 3: Tone decoder signal timing showing *tone_present* flag and action of clock-enable

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
sincos16.vhd	SIN/COS look-up table
dds16.vhd	16-bit DDS component
ddc16.vhd	16-bit Digital Down Converter
iir_biquad.vhd	IIR filter
lpf.vhd	Parameterizable dual-channel low pass I/Q filter
tone_dec.vhd	Top-level component
tone_dec_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. sincos16.vhd
2. dds16.vhd
3. ddc16.vhd
4. iir_biquad.vhd
5. lpf.vhd
6. tone_dec.vhd
7. tone_dec_bench.vhd

The VHDL testbench instantiates the Tone Decoder component and also a separate DDS that provides the source input signal. The tone centre frequency of the source DDS may be modified by adjusting the phase increment accordingly.

In the example test provided, the Tone Decoder is configured to use filter type '0' with the magnitude threshold set to 1024. The system clock period is set to 100MHz.

The tone frequency is set to 3MHz and the test sequences through a series of tones in 2 us intervals. The tone sequence is: 1MHz, 2MHz, 3MHz, 4MHz, 5MHz, 4MHz, 3MHz ... etc. The resultant output waveforms for the test is shown in Figure 4.

Notice that the output magnitude peaks when the 3MHz input tone is present in the source signal. There is a delay between the change in input tone and it's detection due to the response time of the filter. Changing the filter type will adjust the time it takes for the decoder to respond to a change in the input signal.

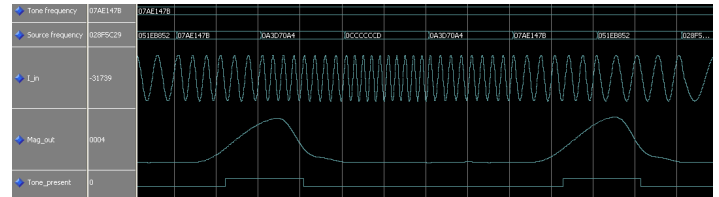


Figure 4: Tone decoder simulation output

The simulation must be run for at least 50 us during which time the output magnitude samples will be captured to a text file called *tone_dec_out.txt*.

Synthesis

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

- tone_dec.vhd
 - ddc16.vhd
 - dds16.vhd
 - sincos16.vhd
 - lpf.vhd
 - iir_biquad.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.

Note that setting the parameter *use_complex* to 'false' will result in a saving of hardware multiplier components. In addition, a lower specification DDS component is also available if a reduction in block RAM resources is preferred³.

Trial synthesis results are shown with the generic parameters set to: dithering = true, seed = 0x14FFDE78, use_complex = false, filter_type = 0, threshold = 1024.

Resource usage is specified after Place and Route.

³ Please contact ZIPcores for details regarding the various DDS options available.

VIRTEX 5

<i>Resource type</i>	<i>Quantity used</i>
Slice register	438
Slice LUT	864
Block RAM	15
DSP48	8
Clock frequency (worst case)	100 MHz
Clock frequency (best case)	142 MHz

STRATIX III

<i>Resource type</i>	<i>Quantity used</i>
Register	367
ALUT	475
Block Memory bit	491742
DSP block 18	18
Clock frequency (worse case)	106 MHz
Clock frequency (best case)	151 MHz

Revision History

<i>Revision</i>	<i>Change description</i>	<i>Date</i>
1.0	Initial revision	29/10/09