Key Design Features

- Synthesizable, technology independent soft IP Core for FPGA, SoC and ASIC
- Supplied as human readable VHDL (or Verilog) RTL source code
- 16-bit signed input and output data samples
- Accepts either complex or real inputs
- Precision Digital Down Converter with 80 dB SFDR
- DDS Frequency resolution of F_S / 2^{32}
- DDS Phase resolution of 2π / 2^{12}
- Programmable tone centre frequency and detection bandwidth
- Choice of different low pass filter responses

Applications

- Touch tone decoding (e.g. DTMF tones)
- Precision frequency monitoring and control
- Security systems and remote control units
- Complex digital down conversion
- Basic building block in FSK / OOK / ASK demodulation

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
gain	Internal gain setting (compensates for low amplitude input signals)	integer	0: x 1 1: x 2 2: x 4 3: x 8
dithering	Enable phase dithering in DDS component	boolean	TRUE / FALSE
seed	Seed for random number generator in DDS component	std_logic vector	0 < 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 (Bandwidth setting)	integer	0: min B/W 1: narrow B/W 2: wide B/W 3: max B/W

Block Diagram

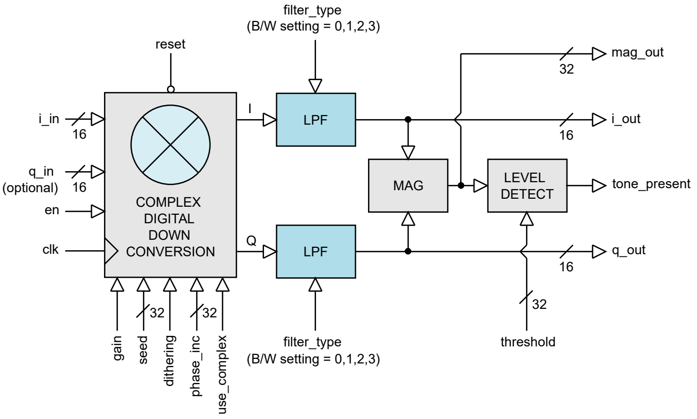

Figure 1: Tone decoder basic architecture

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
threshold [31:0]	in	Tone detection threshold (integer in range 0 to 2^{31})	unsigned number
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 [31:0]	out	Magnitude of complex output	data
tone_present	out	Tone present flag	high

General Description

The TONE_DEC IP Core is a precision tone decoder with the ability to support either real or complex data samples. Samples are first mixed-down 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 80 dBs (with phase dithering) and a theoretical SNR of approximately 100 dBs.

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 input *threshold* signal is a 32-bit unsigned integer that generates the *tone_present* flag when the magnitude is greater than this value. The threshold signal may be programmed in real time.

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:

$$\Phi_{INC} = \left(F_{OUT} * 2^{32}\right) / F_S$$

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:

$$F_{MIN} = F_S / 2^{32} \quad , \qquad F_{MAX} = F_S / 2$$

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).

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 with a shorter 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)	0.03 * (F_S / 2)	50 samples

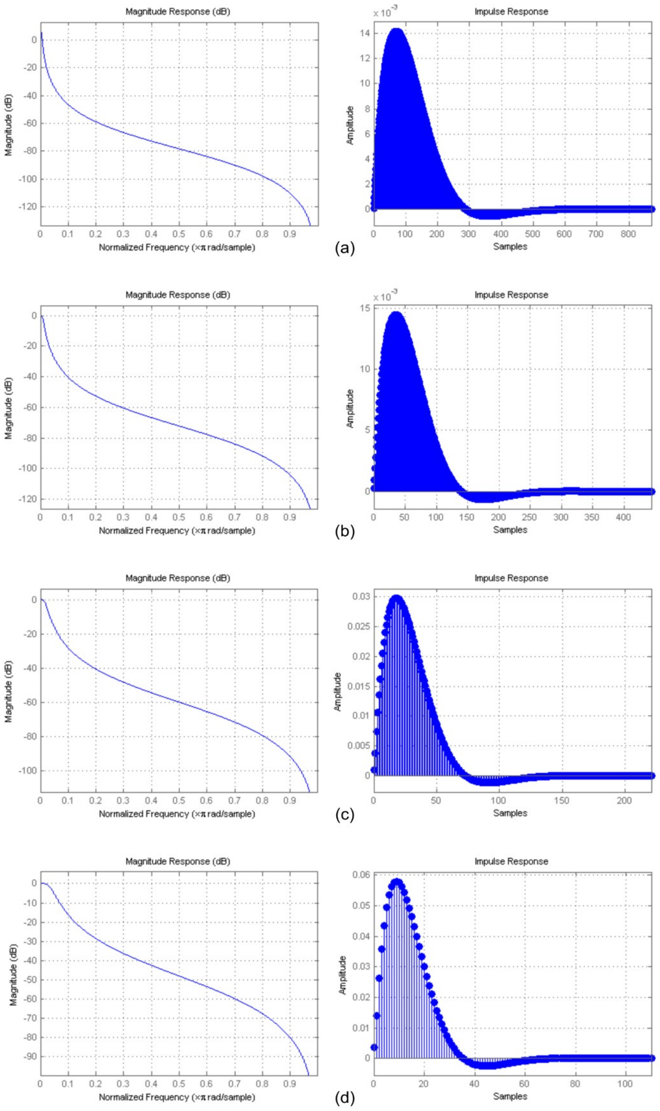

Figure 2: Low-pass filter responses. The -3dB cutoff points are: (a) 0.005, (b) 0.001, (c) 0.02 and (d) 0.03 rads/sample

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 generic *gain* parameter may be adjusted accordingly.

Functional Timing

Figure 3 shows the operation of the tone decoder during normal operation. In this particular example, the *threshold* has been set to 0x04000000 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 detection circuit. This may be necessary depending on the input signal dynamic range and the chosen filter type. The waveforms also show the action of the clock-enable signal *en*.

Samples are clocked on a rising clock-edge when *en* is high. (Note that in order to keep the diagram simple, only the top 16-bits of the *threshold* an *mag_out* signals are shown. In practice, these are 32-bit unsigned numbers as of rev 1.4 of the source code).

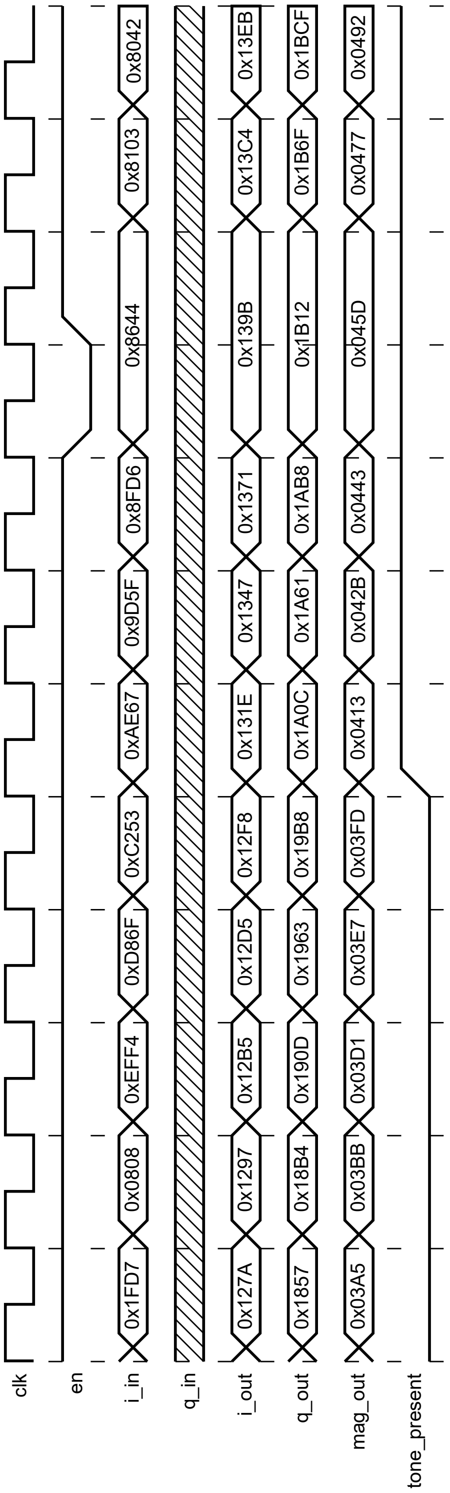

Figure 3: Tone decoder signal timing showing tone_present flag and the action of clock-enable.

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	SINE / COSINE look-up table
dds16.vhd	16-bit DDS component
ddc16.vhd	16-bit digital down converter
iir_biquad.vhd	IIR filter core component
lpf.vhd	Dual-channel low-pass I/Q filter with selectable B/W setting
tone_dec.vhd	Top-level component
tone_dec_bench.vhd	Top-level test bench

Functional Testing

An example VHDL test bench is provided for use in a suitable hardware simulator. The compilation order of the source code is the same as that outlined in the source file description (above).

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

In the example test provided, the tone decoder is configured to use filter type '0' with the magnitude threshold set to 2^{17}. The system clock period is set to 100 MHz.

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

The simulation must be run for at least 1 ms during which time the output magnitude samples will be captured to a text file called *tone_dec_out.txt*.

Development board Testing

The TONE_DEC IP Core was also implemented on a low-cost Zedboard evaluation platform to ensure correct operation. The list of equipment used was as follows:

- Xilinx / Digilent Zedboard
- Zipcores ZIP-FMC-DSP Rev. C mezzanine card[1]
- Tektronix AFG31000 arbitrary function generator
- Tektronix MDO3014 mixed-domain oscilloscope
- 50 Ohm BNC/SMA coaxial cables

One of the ADC input channels of the Zipcores FMC-DSP card was used to sample the source signal into the Zedboard (Zynq SoC) platform. The source signal was a 1V pk-pk sine wave sweep from 1MHz to 5MHz. The TONE_DEC IP Core was configured to detect the tone of interest at 3MHz. The magnitude *threshold* setting was adjusted to get a good clean *tone_present* signal at 3 MHz.

[1] Zipcores' ZIP-FMC-DSP card with dual high-speed ADC and DAC. Please see: www.zipcores.com/fmc-dsp-mezzanine-card.html for more information.

Both the input sine wave sweep signal and the *tone_present* signal were monitored on the scope to determine correct operation. The *tone_present* signal was routed to one of the Zedboard PMOD header pins on the board. The photo below (Figure 4) shows the general bench setup with the scope traces and the *tone_present* flag asserted high when the 3MHz tone is detected in the sweep.

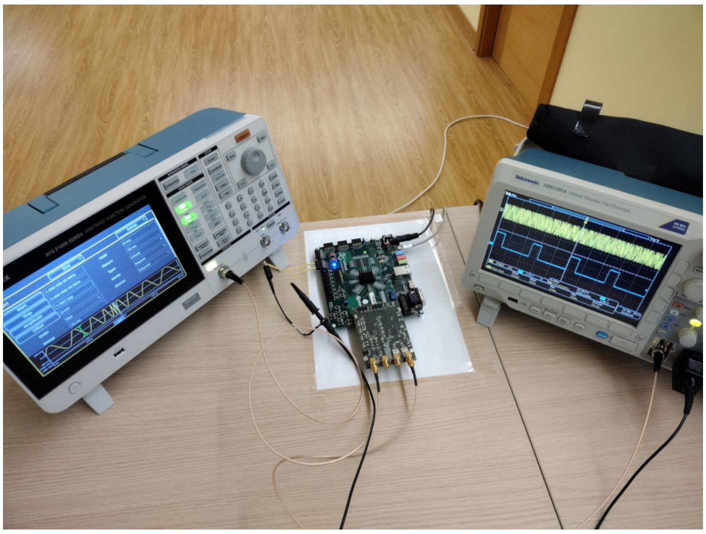

Figure 4: Photo of bench setup with the Zedboard and Zipcores' ZIP-FMC-DSP mezzanine card connected to the FMC slot

Synthesis and Implementation

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 IP Core is designed to be technology independent, however, as a benchmark, synthesis results have been provided for the AMD / Xilinx® 7-series FPGAs and SoCs. Synthesis results for other devices and technologies can be provided on request.

Note that setting the parameter *use_complex* to 'false' will result in a saving of hardware multiplier components. This is because the Q-channel input signal path is removed in the complex mixing process.

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

Resource usage is specified after place and route.

AMD / XILINX® 7-SERIES FPGAS

Resource type	***A-7***	***K-7***	***V-7***	***V-US+***
Slice Register	307	307	307	307
Slice LUTs	1003	1052	1053	1011
Block RAM	0	0	0	0
DSP	18	18	18	18
Occupied Slices	319	314	327	179 (CLB)
Clock freq (approx)	100 MHz	150 MHz	200 MHz	300MHz+

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision.	29/10/2009
1.1	Updated block diagram.	17/03/2010
1.2	Modified the DDS LUT to use 12-bits internally in order to reduce Block RAM usage. Updated synthesis results.	29/12/2011
1.3	Added the gain generic parameter and optimized the IIR filter for speed.	25/02/2015
1.4	Made the threshold parameter fully programmable. Made the magnitude output 32-bits for extra sensitivity. Updated synthesis results for Xilinx 7-series devices.	17/03/2023