Summary - Peak Detector

Name	peak_detector
Version	v1.3
Release Date	Feb 2018
Component Library	ocpi.training.components
Workers	peak_detector.rcc, peak_detector.hdl
Tested Platforms	c7-x86_64, linux-x13_3-arm, xsim, isim, Matchstiq-Z1(PL)(Vivado 2017.1 and ISE 14.7)

Functionality

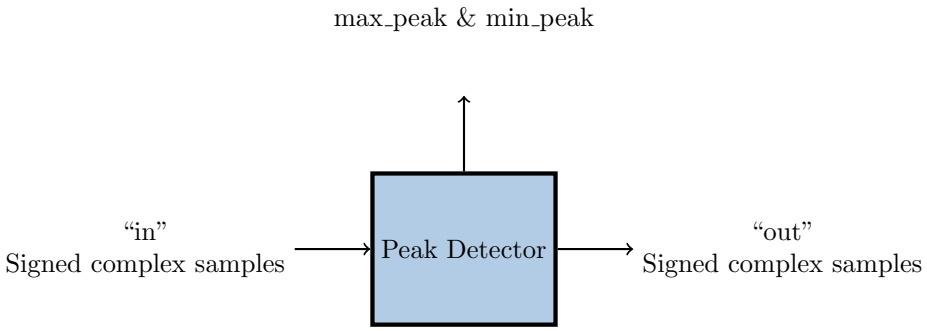
The Peak Detector worker utilizes the OCPI *iqstream_protocol* for both input and output ports. The *iqstream_protocol* defines an interface of 16-bit complex signed samples. The worker calculates the maximum and minimum peaks of the I/Q data arriving at its input port, and passes the input data through to the output port.

The Peak Detector worker uses two local variables to keep track of the maximum and minimum peak amplitudes. To ensure the peaks are detected correctly, the variable used to keep track of the maximum peak is initialized to the most negative value represented in a signed 16-bit number (-32768), and the minimum peak is initialized to the most positive value represented in a signed 16-bit number (32767).

Upon completion, the Peak Detector returns the most positive I or Q sample value with the `max_peak` property and the most negative with the `min_peak` property. *This is not the value of the vector represented by I and Q, but simply the max/min value of the I and Q samples taken independently.*

Block Diagrams

Top level



State Machine

No finite-state machines (FSM) are implemented by this worker.

Source Dependencies

peak_detector.rcc

- training_project/components/peak_detector.rcc/peak_detector.cc

peak_detector.hdl

- training_project/components/peak_detector.hdl/peak_detector.vhd

Component Spec Properties

Name	OCS	OWD RCC	OWD HDL	Type	Length	Accessibility	Valid Range	Default	Usage
`max_peak`	Property	N/A	N/A	short scalar	N/A	Volatile	-32768 to 32767	N/A	Used to return the maximum amplitude value
`min_peak`	Property	N/A	N/A	short scalar	N/A	Volatile	-32768 to 32767	N/A	Used to return the minimum amplitude value

Worker Properties

N/A

Component Ports

Name	Producer	Protocol	Optional	Usage
in	false	`iqstream_protocol`	false	I(31:16); Q(15:0); DataWidth=32; If (DATA_WIDTH_p < 16), then the data must be signed extended
out	true	`iqstream_protocol`	false	I(31:16); Q(15:0); DataWidth=32; If (DATA_WIDTH_p < 16), then the data must be signed extended

Worker Interfaces

peak_detector.hdl

Type	Name	DataWidth	Advanced	Usage
StreamInterface	in	32	N/A	Signed complex samples
StreamInterface	out	32	N/A	Signed complex samples

Control Timing and Signals

The Peak Detector worker uses the clock from the Control Plane and standard Control Plane signals.

Performance and Resource Utilization

peak_detector.hdl

Table entries are a result of building the worker with default parameters:

Device	Registers	LUTs	Fmax	Memory/Special Functions	Design Suite
Zynq XC7Z010-3-CLG400	159	249	516.627 MHz	N/A	ISE 14.7
Zynq XC7Z020-1-CLG484	162	153	345.781 MHz	N/A	Vivado 2017.1

Test and Verification

A single test case is implemented to validate the peak_detector component. An input file is generated (via *generate.py*) containing complex signed 16-bit samples with a tone at 13 Hz. The input data is passed through the worker, so the output file should be identical to the input file. The worker measures the minimum and maximum amplitudes found within the complex data stream. These values, reported as properties, are compared with min/max calculations performed during verification (*verify.py*).

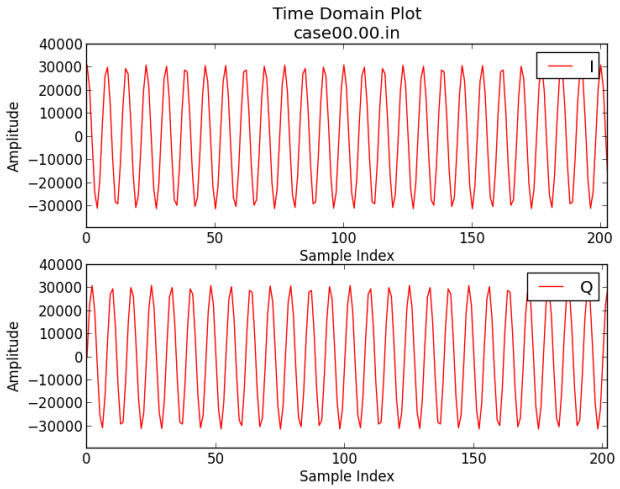


Figure 1: Input Time Domain Tone

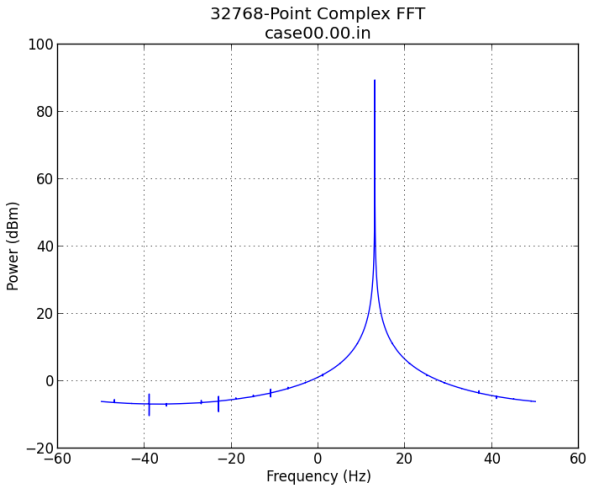


Figure 2: Input Frequency Domain Tone

The output file is first checked that the data is not all zero, and is then checked for the expected length of 32,768 complex samples. Once these quick checks are made the minimum and maximum values are calculated from the file and then compared with the UUT reported values. Figures 3 and 4 depict the output of the Peak Detector worker, where the time domain plot displays the first 200 complex samples.

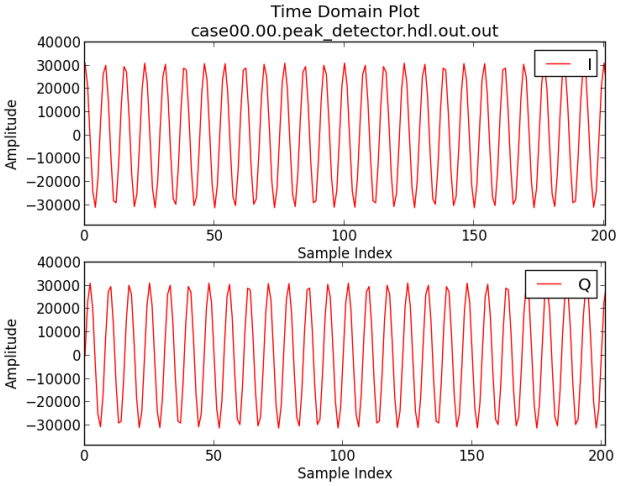


Figure 3: Output Time Domain Tone

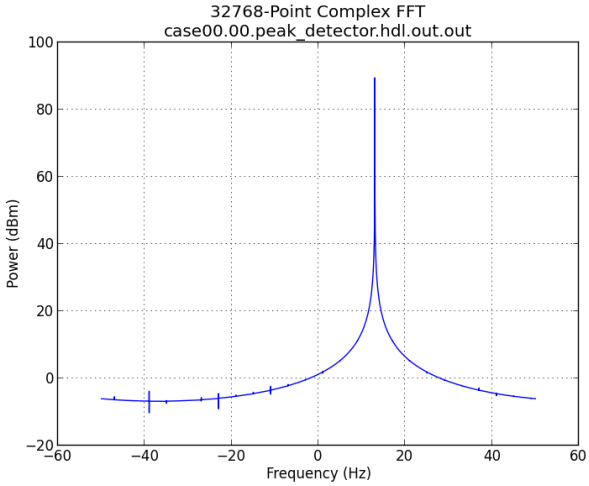


Figure 4: Output Frequency Domain Tone