Summary - AGC Real

WARNING: Run-time failures have been observed for several unit tests and applications on PCIe-based platforms. The work around requires modifying the **buffersize** attribute of the PL to PS (egress) boundary connection, as defined in the OAS. An example for modifying the OAS for executing on PCIe-based platforms is provided in a below section.

Name	agc_real
Worker Type	Application
Version	v1.4
Release Date	February 2018
Component Library	ocpi.assets.util_comps
Workers	agc_real.hdl
Tested Platforms	xsim, isim, modelsim, alst4, ml605, ZedBoard(PL), Matchstiq-Z1(PL)

Functionality

The Automatic Gain Control (AGC) Real component inputs real signed samples, drives the amplitude of the input data to a reference level, and outputs real signed samples.

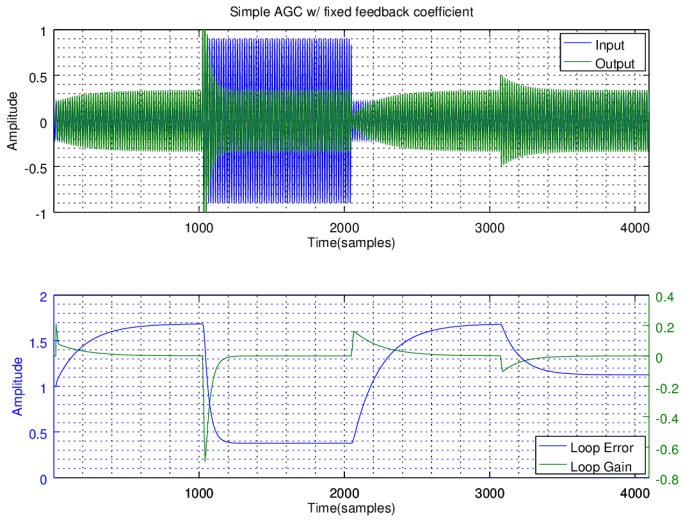


Figure 1: MATLAB AGC implementation with ref=0x1B26 and mu=0x144E

Worker Implementation Details

agc_real.hdl

The response time and output level of the circuit are programmable, as is the ability to update/hold the gain differential used in the feedback loop. The size of the averaging window used for peak detection is build-time programmable using the `AVG_WINDOW_p` parameter, which is recommended to be a power-of-two to enable hardware division implementation with shift registers.

The `ref` property controls the desired output amplitude, while the `mu` property controls the AGC time constant, thus determining the response time of the circuit.

This implementation uses three multipliers to process input data at the clock rate - i.e. this worker can handle a new input value every clock cycle. This circuit will produce output one clock cycle after each valid input, but the input-to-output latency is actually three valid clock cycles.

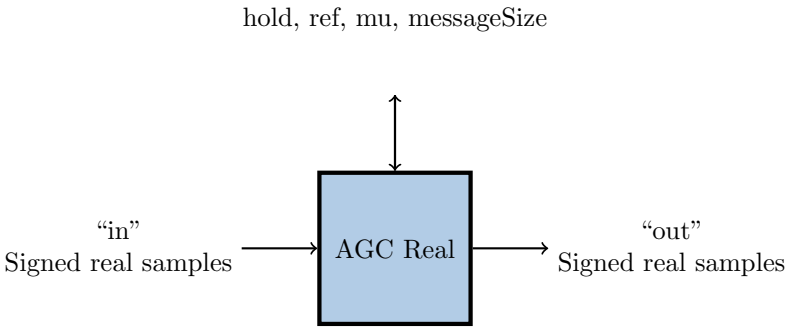
The AGC Real worker utilizes the OCPI *rstream_protocol* for both input and output ports. The *rstream_protocol* defines an interface of 16-bit real signed samples. The `DATA_WIDTH_p` parameter may be used to reduce the worker's internal data width to less than 16-bits.

Theory

The circuit is based upon Richard G. Lyons' "Understanding Digital Signal Processing, Third Edition" Automatic Gain Control (AGC) circuit found on Page 783. The text may also be found online here: DSP-Tricks-A simple way to add AGC to your communications receiver design. Lyons' circuit in Figure 13-76a implements the AGC function with a feedback loop on y(n) that consists of a magnitude operation to remove the sign, a comparator against the reference level "ref", a multiplier that uses "mu" to control the amplitude of the feedback signal (and thus the response time), and finally an accumulator. This implementation uses a peak detector in place of Lyons' simple magnitude operation. From Lyons: "The process is a nonlinear, time-varying, signal-dependent feedback system. As such, it's highly resistant to normal time-domain or z-domain analysis. This is why AGC analysis is empirical rather than mathematical ..."

Block Diagrams

Top level



State Machine

Only one finite-state machine (FSM) is implemented by this worker. The FSM supports Zero-Length Messages.

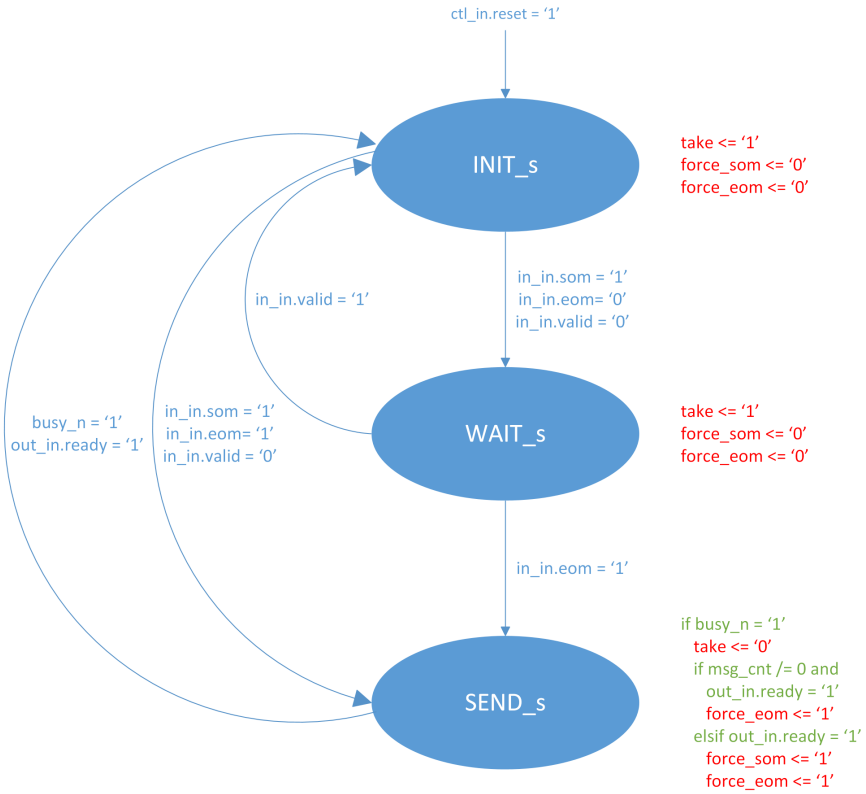


Figure 2: Zero-Length Message FSM

Note: In future releases this finite-state machine will be replaced with a register-delay based mechanism, currently exemplified in the dc offset filter

Source Dependencies

agc_real.hdl

- assets/components/util_comps/agc_real.hdl/agc_real.vhd
- assets/hdl/primitives/util_prims/util_prims_pkg.vhd
  assets/hdl/primitives/util_prims/agc/src/agc.vhd

Component Spec Properties

N/A

Worker Properties

agc_real.hdl

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
Property	DATA_WIDTH_p	UShort	-	-	Readable, Parameter	1-16	16	Worker internal non-sign-extended data width
Property	AVG_WINDOW_p	UShort	-	-	Readable, Parameter	4-256	16	Length of the averaging buffer; should be a power of two
Property	hold	Bool	-	-	Readable, Writable	Standard	false	Hold disables the gain differential feedback circuit, thus maintaining the current gain
Property	ref	UShort	-	-	Readable, Writable	1 to $2^{\texttt{DATA\_WIDTH\_p}} - 1$	0x3FFF	Desired output amplitude expressed in percentage of full scale expected peak value in rms
Property	mu	UShort	-	-	Readable, Writable	1 to $2^{\texttt{DATA\_WIDTH\_p}} - 1$	N/A	Feedback coefficient used to control the response time of the circuit; expressed as mu*fullscale
Property	messageSize	UShort	-	-	Readable, Writable	8192	8192	Number of bytes in output message

Component Ports

Name	Producer	Protocol	Optional	Advanced	Usage
in	false	rstream_protocol	false	-	Real signed samples
out	true	rstream_protocol	false	-	Real signed samples

Worker Interfaces

agc_real.hdl

Type	Name	DataWidth	Advanced	Usage
StreamInterface	in	16	ZeroLengthMessages=true	Signed real samples
StreamInterface	out	16	ZeroLengthMessages=true	Signed real samples

Control Timing and Signals

The AGC Real worker uses the clock from the Control Plane and standard Control Plane signals.

Worker Configuration Parameters

agc_real.hdl

Table 1: Table of Worker Configurations for worker: agc_real

Configuration
0

Performance and Resource Utilization

agc_real.hdl

Table 2: Resource Utilization Table for worker: agc_real

Configuration	OCPI Target	Tool	Version	Device	Registers (Typ)	LUTs (Typ)	Fmax (MHz) (Typ)	Memory/Special Functions
0	zynq	Vivado	2017.1	xc7z020clg484-1	375	411	N/A	DSP48E1: 3
0	virtex6	ISE	14.7	6vlx240tff1156-1	389	426	171.91	DSP48E1: 3
0	stratix4	Quartus	17.1.0	EP4SGX230KF40C2	694	345	N/A	DSP18: 6

Test and Verification

WARNING: Run-time failures have been observed for several unit tests and applications on PCIe-based platforms. The work around requires modifying the **buffersize** attribute of the PL to PS (egress) boundary connection, as defined in the OAS.
The OAS XML must be changed to resolve this issue for the unit tests as shown below:

```
<Connection>
 <Port Instance="last_PL_worker" Name="out"/>
 <Port Instance="first_PS_worker" Name="in" Buffersize="8192" Buffercount="4"/>
</Connection>
```

A single test case is implemented to validate the AGC Real component. An input file is generated with a single tone at Fs/16 Hz, where Fs = 100 Hz, but applies 20% of the maximum amplitude to the first quarter of the file, 90% maximum amplitude to the second quarter of the file, 20% maximum amplitude to the third quarter of the file, and 30% maximum amplitude to the fourth quarter of the file. The real waveform is then scaled to fixed-point signed 16-bit integers. Time and frequency domain plots may be viewed in Figures 3 and 4 below, respectively.

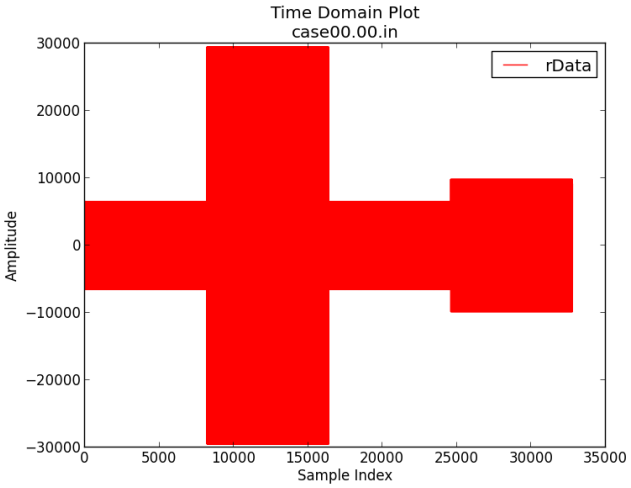


Figure 3: Time Domain Tone

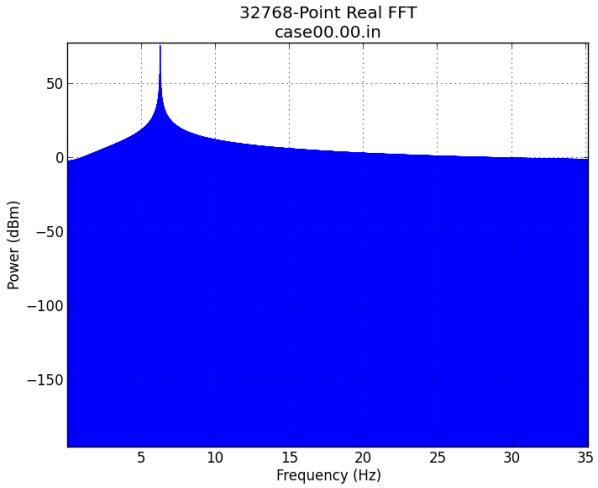


Figure 4: Frequency Domain Tone

For verification, the output file is first checked that the data is not all zero, and is then checked for the expected length of 32,768 real samples. Once these quick checks are made a floating-point python implementation of the AGC is performed on the input data, which is then compared sample-by-sample to the output data. Figures 5 and 6 depict the output of the AGC Real worker.

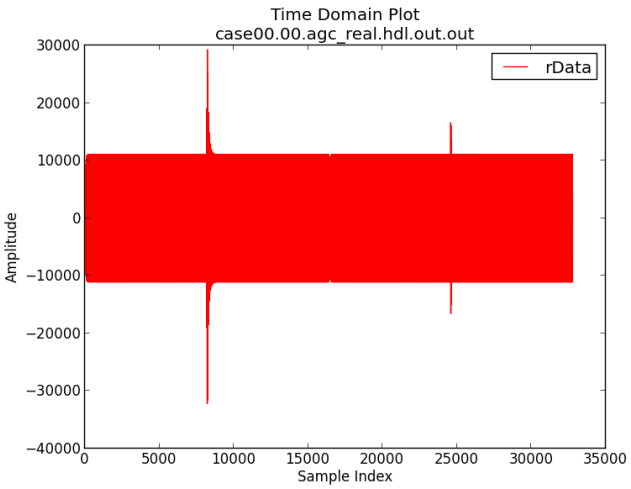


Figure 5: Time Domain Tones with AGC

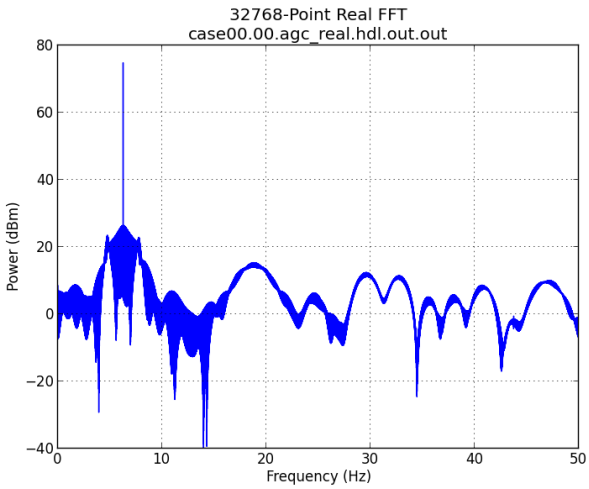


Figure 6: Frequency Domain Tones with AGC

References

(1) Richard G. Lyons. *Understanding Digital Signal Processing.* Third Edition. Pearson Education, Inc., Boston. 2001.

(2) Richard G. Lyons. (2011, March 29). *A simple way to add AGC to your communications receiver design.* Retrieved from http://www.embedded.com/design/other/4214571/A-simple-way-to-add-AGC-to-your-communications-receiver-design-.