Application Note:

Playback function and equalizer on SDR14

Table of Contents

Introduction **2**
Playback in standard firmware **2**
Activating Playback 2
AWG check function 3
Specification 3
Example code 3
API Commands 3
Advanced playback example code in SDR14 Development Kit **4**
About SDR14 Development Kit 4
Block diagram 4
Specification 5
Tunable delay line 5
Gate function 5
Calibrating equalizer introduction 6
Gain setting 6
MATLAB example for calibrating the equalizer 7
MATLAB example for starting playback 8
API commands 8

Introduction

The SDR14 has two 14-bit output channels capable of an update rate of 1600 MSPS each and two 14-bit input channels at 800 MSPS each. This document provides an instruction on how to connect the analog inputs to the analog outputs inside the SDR14 to create a playback function.

The playback functionality is available in two versions; one that is included in the standard FPGA firmware delivered with SDR14, and one that is available through the custom firmware development tool **SDR14 Development Kit**. The playback is a method for playing an echo of the recorded signal with minimum latency. The standard firmware of SDR14 contains a fundamental playback function, whereas the **SDR14 Development Kit** contains source code for creating an advanced custom playback application.

SDR14 Development Kit is ordered separately.

Playback in standard firmware

Activating Playback

The standard firmware contains a playback mode which is activated through user register. By writing a logical “1” to bit number 0 in user register 0 (address 10240), the playback function is activated. This means disconnect the arbitrary waveform generator (AWG) from the DAC and loop back ADC-data directly.

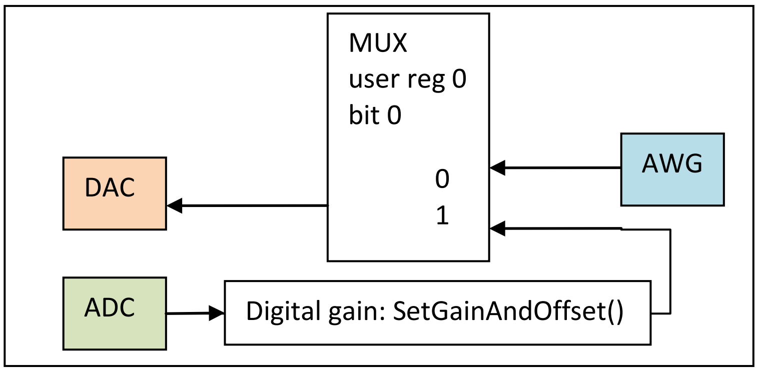

Figure: Playback ADC data to DAC.

Digital gain is 1, that is full scale signal on the ADC is replayed as full scale analog signal out from the DAC. The full scale analog output power from the DAC is lower than the full scale analog input power to the ADC. This may be compensated for by using the SetGainAndOffset() command. By setting a high value on the digital gain, the loss in analog signal level is compensated for. Note that the analog input signal range is reduced to match the output signal power.

The sample rate through the playback application is 800 MSPS. The ADC runs at 800 MSPS, whereas the DAC runs at an update rate of 1600 MHz. The ADC data is interpolated to 1600 MSPS by outputting same data twice. This interpolation method minimizes the loop delay. In this mode the DAC will operate as an 800 MSPS DAC.

AWG check function

There is also an AWG waveform check function that can be used for verifying the AWG signal. The function uses the digital recorder to verify the output from the AWG. Instead of writing data from the ADC into the DRAM, data from the AWG is written directly to the DRAM. Note that neither ADC nor DAC are involved in this operation.

By writing a logical “1” to bit number 1 in user register 0 (address 10240), the AWG check function is activated.

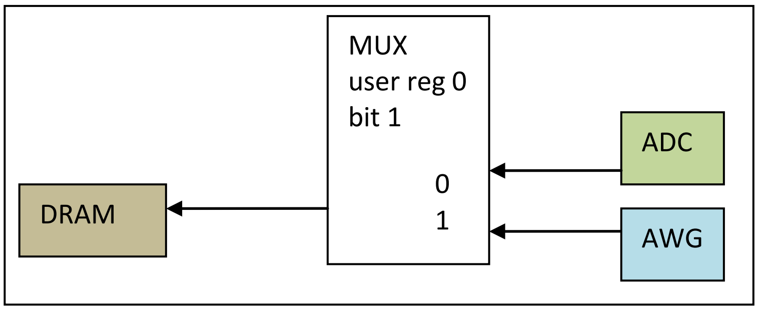

Figure: Checking AWG by using digitizer digital recorder.

Specification

• Data rate	800 MSPS
• Time delay input to output	215 ns typically

Note that the minimum delay time is measured from ADC input to DAC output. It may thus vary between individual units due to delay in analog components. Future updates of the firmware may add or remove pipelining steps in the data path, which may affect this value.

Example code

MATLAB example code:

```
    bypass_adctodac = 0; % 1 = loop ADC-data to DAC output, 0 = normal operation
    interface_ADQ('writeregister',[10240 hex2dec(‘FFFFFFFC’) bypass_dactoadc*2^1 +
bypass_adctodac*2^0], boarded);
```

C example code:

```
    SDR14_WriteRegister(adq_cu, 1, 10240, 0xFFFFFFFC, ((bypass_dactoadc << 1) |
(bypass_adctodac << 0)));
```

API Commands

The software functions are described in the ADQ API user’s guide, which is included in the ADQ software development kit that was delivered with the SDR14. See:
SP Devices\Documentation\08-0214_ADQ-API_ug.pdf

Advanced playback example code in SDR14 Development Kit

About SDR14 Development Kit

SDR14 Development Kit is a tool for designing custom FPGA firmware for the SDR14. The **SDR14 Development Kit** contains a XILINX ISE design project, which includes a number of application examples in Verilog. The presented playback function is one of the examples in the **SDR14 Development Kit**. The function is delivered with a default configuration, which is described here. However, since the source code is included, it may be altered by the user to adapt to a specific application. There is software source code that shows how to operate the block, which also may be altered for a specific application.

The **SDR14 Development Kit** requires skills in Verilog or VHDL programming languages. (All examples are written in Verilog only). It also requires deep knowledge of advanced FPGA design. The user interface requires programming skills in C/C++ or MATLAB.

Block diagram

The system contains a loop from the ADC to the DAC via a delay line, a timing gate and filtering. The loop length is set by a variable delay line from the receiver to the transmitter. The transfer function can be adjusted by an equalizer. The sample rate of the ADC is 800 MSPS whereas the update rate of the DAC is 1600 MHz so a 2x interpolation block is included. The gain through the signal chain can be set to a target value which is included in the equalizer calibration. The gate function enables a rectangular window function on the playback signal. In this way, a specific pulse may be selected. The DAC output is then silent in the periods before and after the wanted pulse.

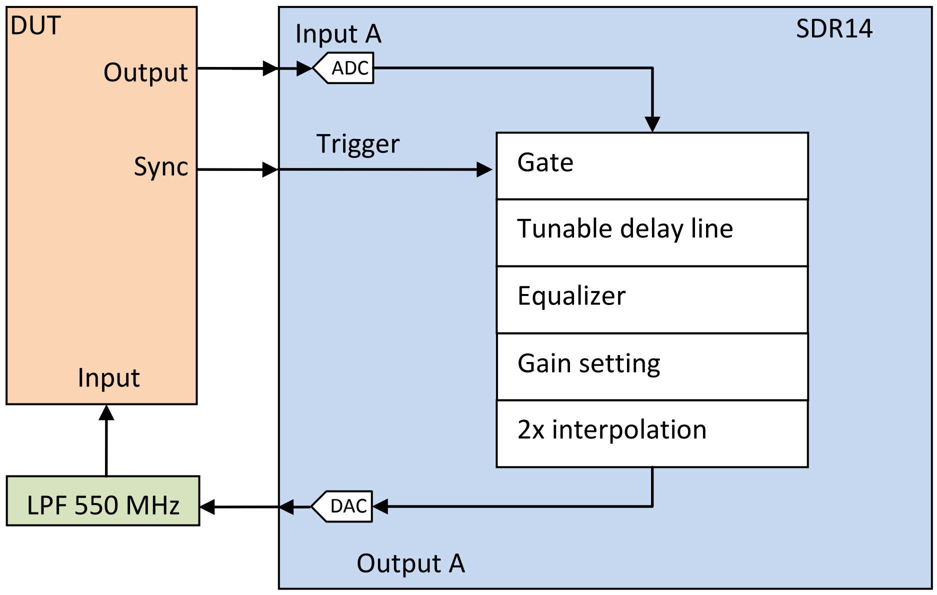

Figure: Block diagram of advance playback in SDR14 Development Kit. Connection to external equipment (DUT) is included. A reconstruction filter at the DAC output is recommended.

Specification

This block is available as source code in an example script in the **SDR14 Development Kit**. It is thus possible to change many of the default parameters.

• Data rate	800 MSPS
• Minimum time delay	500 ns typically
• Maximum time delay	41 us
• Gate trigger hold off max	327 us
• Gate length max	327 us
• Granularity (All time settings)	5 ns

Note that the minimum delay time is measured from ADC input to DAC output. It may thus vary between individual units due to delay in analog components. Future updates of the firmware may add or remove pipelining steps in the data path, which may affect this value.

Tunable delay line

The tunable delay line is a FIFO with programmable length. The minimum latency from the ADC input to DAC output is in the range of 500 ns. The delay line length is up to 41 us, set in steps of 5ns.

Gate function

The gate function is used for getting a quiet output and only transmitting the signal of interest. The gating is controlled by a trigger and the timing of the window is set by the parameters trigger hold-off and window size.

The ADC data is captured continuously but as long as the gate is closed, the data is blocked from entering the delay line. Instead, data is set to 0. After a trigger and trigger hold-off period, the gate is opened and ADC data can pass through to the delay line. The gate length determines when to close the gate.

It is also possible to bypass the gate and continuously send data from the ADC through to the DAC.

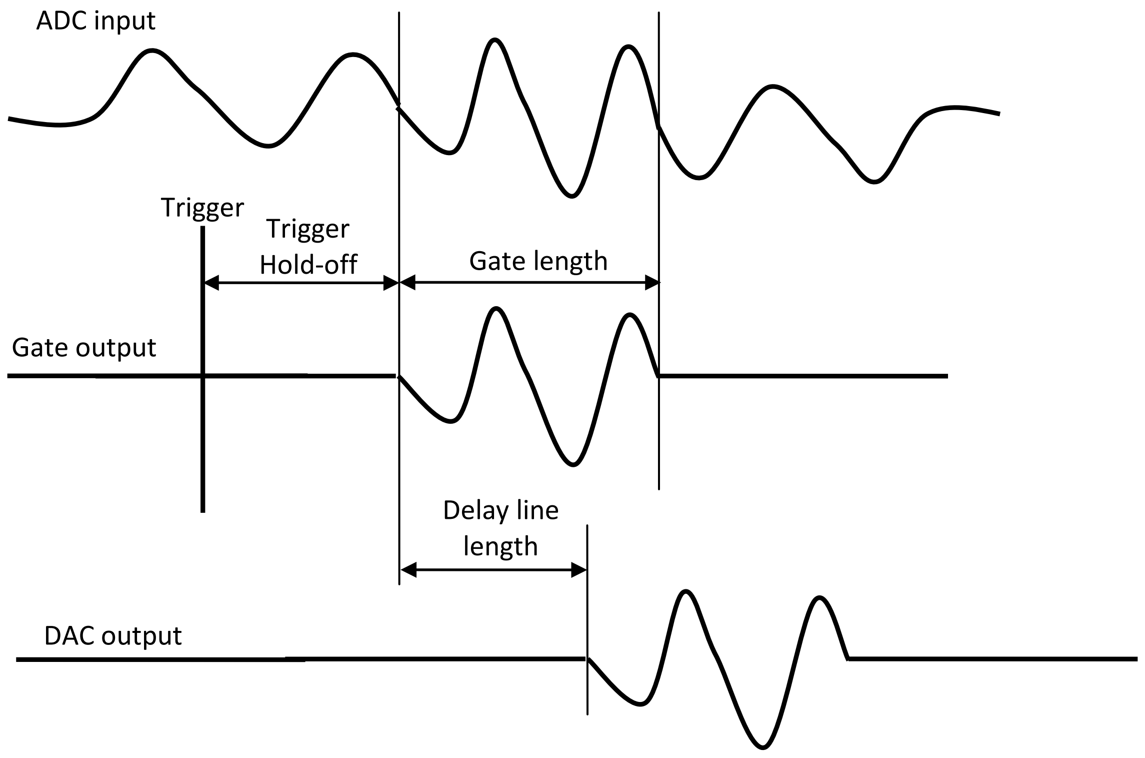

Figure: Timing diagram for the gating function.

Calibrating equalizer introduction

The equalizer compensates for the non-ideal transfer function of the SDR14. The equalizer is calibrated by connecting the output A of the SDR14 to the input of the input A of SDR14 and applying a calibration sequence. Repeat for channel B. One example of how to calibrate an equalizer is illustrated in a MATLAB example file, see below.

Notes regarding equalizer calibration:

- The equalized frequency band is between 5 MHz and 355 MHz, frequencies outside this band are ignored by the equalization algorithm.
- The DAC should have a low-pass reconstruction filter connected externally (both during calibration and normal playback operation). This filter should pass the entire signal band of interest, and attenuate the image centered around 1600 MHz. In the example, a 550 MHz, 5th order, Cauer low-pass filter was used.
- The calibration will diverge if the transfer function between DAC and ADC is too extreme in its frequency characteristic. The present equalizer successfully removes bumps of at about 6-10 dB from the amplitude characteristic.
- The method in the example is only an example of a calibration algorithm.

Gain setting

The target gain through the signal chain is set as an input parameter to the equalizer calibration procedure.

MATLAB example for calibrating the equalizer

A MATLAB example on how to calibrate an equalizer is included. The script can be translated into a C/C++ example, see section “API Commands”.

sdr14_echo_calibrateeq.m

This script calculates equalizer filters for both channel A and B, and then tests the result in hardware. The script saves parameters for the two equalizer filters in a file called *savedequalizer.mat*, which is later used for operating the playback function. It also produces plots for manual inspection of the result. A *ReadMe.txt* file with more instructions is delivered with the example.

The figures below illustrate calibration of the equalizer for a DUT with an open stub. An impulse is sent from the DAC to the DUT and back in to the ADC. The result with and without equalizer are plotted. The echo from the open stub is removed by the equalizer. The frequency response with equalizer is flat.

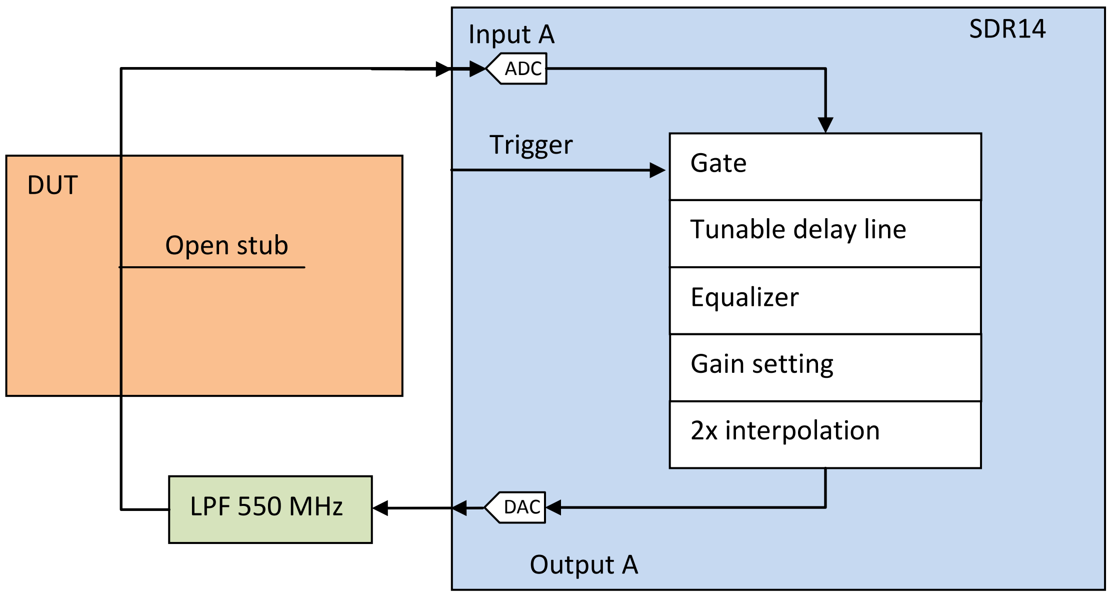

Figure: Example on calibrating the equalizer including DUT with an open stub that causes reflection.

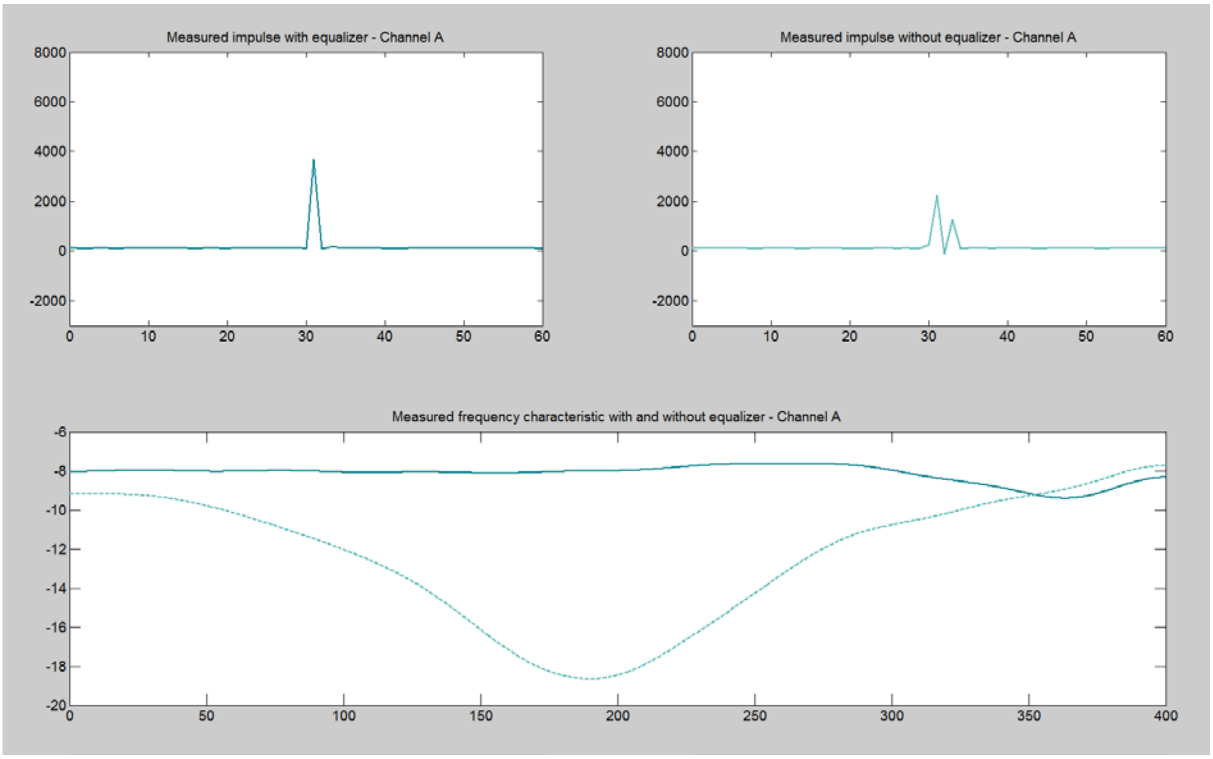

Figure: Calibration result. After the equalizer, the echo in the stub is removed and the frequency response is flat.

MATLAB example for starting playback

One included MATLAB scripts configures the playback SDR14 unit

sdr14_echo_setupplayback.m

This script sets up all the various parameters of the playback system. Various functionalities, such as interpolation, equalizer and gating can be turned on and off, and reconfigured, by setting various variables in this script and then running it. When using an equalizer for playback, the script assumes that you have run *sdr14_echo_calibrateeq.m* beforehand, which creates a file called *savedequalizer.mat* with the calculated equalizer coefficients, which the *sdr14_echo_setupplayback.m* script then uploads to the digitizer every time it is run.
The script can be translated into a C/C++ example, see section "API Commands".

API commands

The control of the playback is done by standard API command. The example may be translated to a C/C++ example in these steps (this strategy is valid for all MATLAB scripts):

- Look in the sdr14_echo_setupplayback.m for calls to *interface_ADQ.m*.
- The first keyword in the call is the command in the C/C++ interface. For example *interface_ADQ('writeregister'...)* is based on the C/C++ function *WriteRegister*.
- The keyword (for example *WriteRegister*) is found in the ADC API user's guide, which is a programmer's reference manual.

- In the file *interface_ADQ.m* is code that maps the MATLAB parameters to the arguments in the C/C++ command. This is helpful when building up the C/C++ function calls.

Both the *interface_ADQ.m* and the ADQ API user's guide are included in the ADQ software development kit that was delivered with the SDR14. See:

SP Devices\Documentation\08-0214_ADQ-API_ug.pdf

SP Devices\Matlab\interface_ADQ.m