Red Pitaya

Thesis

Raphael Frey
Noah Hüsser

May 19, 2017
Version 0.0.1

Contents

List of Figures

List of Tables

List of Listings

CHAPTER 1

Introduction

- Rationale (Why?)
- What is the general approach to solve this problem?
- What has been done so far?
- Results of previous work
- What are we going to do?
- What are the contents of this report?

Part I

System Overview

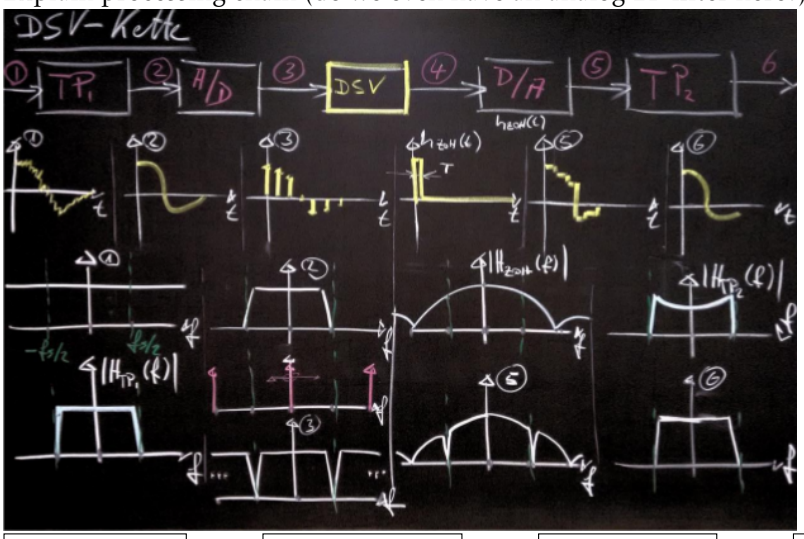
Fundamental question to answer in this part: *What is our system, and what is it good for?*
This information could also go into *Introduction*?
Provide supplementary theoretical background as needed.

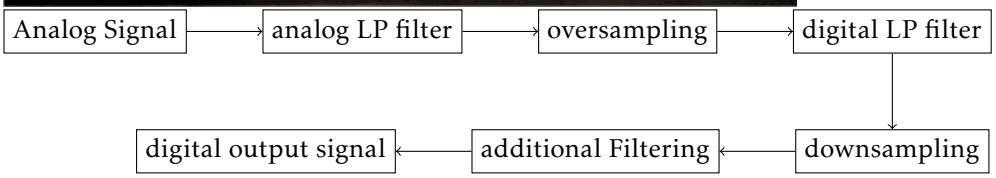
CHAPTER 2

Analog-to-Digital Data Acquisition

- Generic Chapter on some of the basic principles of AD data processing
- Sampling:
  - Sampling in time domain: Multiplication with dirac pulse sequence
  - Frequency doman: Convolution of signal spectrum and dirac pulse spectrum
  - "What is spectrum of dirac pulse sequence?" (dirac pulse sequence)
  - Consequently: Spectrum of sampled signal is repeated for each dirac pulse in the spectrum
  - **Make sure to get distances between pulses as well as heights correct!**
  - Aliasing
  - LP-Filtering (Anti-Aliasing Filter)
  - Potentially mention reconstruction
- Very brief mention of aliasing, low-pass filtering and all that
- Oversampling and Downsampling
- Oversampling: Explain advantage w/r to SNR
- Fancy graphics from Mr. Gut
- How does the Red Pitaya fit into this?
- CIC and FIR filters:
  - Overview (emphasis on CIC)
  - Where to use which, and why?
  - Table/Matrix with advantages and drawbacks for each
  - How does this translate to our system/Why is this important for us?

Explain processing chain (do we even have an analog LP filter here?)





CHAPTER 3

The Red Pitaya Platform

3.1 General Information

General Info about Red Pitaya Project:

- How is the PITA project structured? (logically, license-wise, philosophically)
- Why do we care about this?
- Replacement for scopes (motivation: Why would one use the PITA?)

3.1.1 FPGA

3.1.2 Linux

3.2 Performance and Possible Improvements

- What is the stock solution for downsampling and such? Performance?
- Results of Previous Work
- Consequences for us: Possible paths forward
- System Analysis
- Decision Matrix & Decision
- pgfplots: Ternary diagram?

Part II

Implementation

Implementation can be read independently of previous part, but there should be a red thread from decision to implementation. Deals primarily with design decisions.

Present a diagram with all system components. Then document the components in their respective chapters and sections.

CHAPTER 4

Data Acquisition System

4.1 FPGA

4.2 Kernel Module

CHAPTER 5

Filters

CHAPTER 6

Server

CHAPTER 7

Graphical Front End

Part III

Developer Guide

Documentation for a person who wishes to utilize our system in their work and/or improve upon it?

Make sure to distinguish between *Implementation* and this part. Lines seem a bit blurry to me (R.F.) at the moment (May 19, 2017).

CHAPTER 8

IP Core

Documentation of our FPGA Project (structure, interfaces, registers ...)

CHAPTER 9

Linux

Kernel module, server

CHAPTER 10

Tool Chain

Vivado, Build Box, ARM Linux, TCL, Makefiles, Libs for building server application

Part IV

User Guide

Documentation for the end user. Primarily concerned with the scope front-end.

Part V

Sandbox

CHAPTER 11

Code Listings

11.1 Makefile

Listing 11.1: Makefile Code

```
# ------------------------------------------------------------------------------
# General constants

# Change to adjust the output directory
BUILD = build

# ------------------------------------------------------------------------------
# Constants for the FPGA Core

VIVADO = vivado -nolog -nojournal -mode batch
PART = xc7z010clg400-1

all: all-cores zynq_logger project

all-cores:
	$(VIVADO) -source create_cores.tcl -tclargs $(PART) $(BUILD)/cores
	rm -f vivado*
	rm -f webtalk*

axis_to_data_lanes:
	$(VIVADO) -source create_cores.tcl -tclargs $(PART) $(BUILD)/cores axis_to_data_lanes_v1_0
	rm -f vivado*
	rm -f webtalk*

.PHONY: zynq_logger
zynq_logger:
	cd zynq_logger && make core
```

```
project:
	$(VIVADO) -source make_project.tcl
	rm -f vivado*
	rm -f webtalk*

clean:
	rm -rf $(BUILD)
	rm -rf .Xil
	rm -rf .tmp_versions
```

11.2 Verilog

Listing 11.2: Verilog Code

```
timescale 1 ns / 1 ps

module axi_axis_reader #
(
  parameter integer AXI_DATA_WIDTH = 32,
  parameter integer AXI_ADDR_WIDTH = 16
)
(
  // System signals
  input  wire                      aclk,
  input  wire                      aresetn,

  // Slave side
  input  wire [AXI_ADDR_WIDTH-1:0] s_axi_awaddr,  // AXI4-Lite slave: Write address
  input  wire                      s_axi_awvalid, // AXI4-Lite slave: Write address valid
  output wire                      s_axi_awready, // AXI4-Lite slave: Write address ready
  input  wire [AXI_DATA_WIDTH-1:0] s_axi_wdata,   // AXI4-Lite slave: Write data
  input  wire                      s_axi_wvalid,  // AXI4-Lite slave: Write data valid
  output wire                      s_axi_wready,  // AXI4-Lite slave: Write data ready
  output wire [1:0]                s_axi_bresp,   // AXI4-Lite slave: Write response
  output wire                      s_axi_bvalid,  // AXI4-Lite slave: Write response valid
  input  wire                      s_axi_bready,  // AXI4-Lite slave: Write response ready
  input  wire [AXI_ADDR_WIDTH-1:0] s_axi_araddr,  // AXI4-Lite slave: Read address
  input  wire                      s_axi_arvalid, // AXI4-Lite slave: Read address valid
  output wire                      s_axi_arready, // AXI4-Lite slave: Read address ready
  output wire [AXI_DATA_WIDTH-1:0] s_axi_rdata,   // AXI4-Lite slave: Read data
  output wire [1:0]                s_axi_rresp,   // AXI4-Lite slave: Read data response
  output wire                      s_axi_rvalid,  // AXI4-Lite slave: Read data valid
  input  wire                      s_axi_rready,  // AXI4-Lite slave: Read data ready

  // Slave side
  output wire                      s_axis_tready,
  input  wire [AXI_DATA_WIDTH-1:0] s_axis_tdata,
  input  wire                      s_axis_tvalid
```

```
);

  reg int_rvalid_reg, int_rvalid_next;
  reg [AXI_DATA_WIDTH-1:0] int_rdata_reg, int_rdata_next;

  always @(posedge aclk)
  begin
    if(~aresetn)
    begin
      int_rvalid_reg <= 1'b0;
      int_rdata_reg <= {(AXI_DATA_WIDTH){1'b0}};
    end
    else
    begin
      int_rvalid_reg <= int_rvalid_next;
      int_rdata_reg <= int_rdata_next;
    end
  end

  always @*
  begin
    int_rvalid_next = int_rvalid_reg;
    int_rdata_next = int_rdata_reg;

    if(s_axi_arvalid)
    begin
      int_rvalid_next = 1'b1;
      int_rdata_next = s_axis_tvalid ? s_axis_tdata : {(AXI_DATA_WIDTH){1'b0}};
    end

    if(s_axi_rready & int_rvalid_reg)
    begin
      int_rvalid_next = 1'b0;
    end
  end

  assign s_axi_rresp = 2'd0;

  assign s_axi_arready = 1'b1;
  assign s_axi_rdata = int_rdata_reg;
  assign s_axi_rvalid = int_rvalid_reg;

  assign s_axis_tready = s_axi_rready & int_rvalid_reg;

endmodule
```

11.3 VHDL

```
----------------------------------------------------------------------------------
--
-- comparator.vhd
--
-- (c) 2015
-- L. Schrittwieser
-- N. Huesser
--
----------------------------------------------------------------------------------
--
-- Old descision piece for the trigger units; obsolete
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity comparator is
        generic (
                Width : integer := 14
        );
    port (
        AxDI : in unsigned(Width - 1 downto 0);
        BxDI : in unsigned(Width - 1 downto 0);
        GreaterxSO : out std_logic;
        EqualxSO : out std_logic;
        LowerxSO : out std_logic
    );
end comparator;

architecture Behavioral of comparator is
begin
    process(AxDI, BxDI)
    begin
        GreaterxSO <= '0';
        EqualxSO <= '0';
        LowerxSO <= '0';
        if AxDI > BxDI then
            GreaterxSO <= '1';
        elsif AxDI = BxDI then
            EqualxSO <= '1';
        else
```

```
            LowerxSO <= '1';
        end if;
    end process;
end Behavioral;
```

11.4 TCL

```
# ============================================================================================================
# make_cores.tcl
#
# Simple script for creating and installing all the IPs in a given directory.
# The script must be run from inside the directory it resides.
#
# by Noah Huesser <yatekii@yatekii.ch>
# based on Anton Potocnik, 01.10.2016
# ============================================================================================================

set part_name [lindex $argv 0]
set build_location [lindex $argv 1]
if {[llength $argv] > 2} {
        set core_names [lindex $argv 2]
}

#set cores [lindex $argv 0]
set cores cores

if {$rdi::mode != "batch"} {[puts "Installing cores from $cores into Vivado..."]}

if {! [file exists $cores]} {
        puts "Directory $cores was not found. No cores were installed.";
        return
}

# Generate a the list of IP cores in the $cores directory if we didn't receive names
if {! [info exists core_names]} {
        cd $cores
        set core_names [glob -type d *]
        cd ..
}

#set core_names "axis_to_data_lanes_v1_0";
#set core_names "axis_red_pitaya_adc_v1_0";

# Import Pavel Demin's Red Pitaya cores
foreach core $core_names {
        set argv "$part_name $build_location $core"
        if {$rdi::mode != "batch"} {[puts "Installing $core..."]}
        source scripts/add_core.tcl
        if {$rdi::mode != "batch"} {[puts "============================"]}
}
```

```
entity comparator is
    generic (
        Width : integer := 14
    );
    port (
        AxDI : in unsigned(Width - 1 downto 0);
        BxDI : in unsigned(Width - 1 downto 0);
        GreaterxSO : out std_logic;
        EqualxSO : out std_logic;
        LowerxSO : out std_logic
    );
end comparator;
```

Listing 11.3: Comparator

11.5 Matlab

Listing 11.4: Matlab Code

```
% ------------------------------------------------------------------------------ %
% FILTER DESIGN ITERATIONS
%
% DESCRIPTION
% Designs and showcases various filter chains for evaluation.
%
% AUTHORS:
% Raphael Frey, <rmfrey@alpenwasser.net>
%
% DATE:
% 2017-MAY-12
% ------------------------------------------------------------------------------ %

% Parameter Description
% R: rate decimation
% N: Number of CIC filter stages
% M: differential delay in CIC combs

% Global Input Sampling Frequency: 125 MHz
%
% Desired Target Frequencies:       25 MHz (R =                5)
%                                    5 MHz (R = 5^2       =   25)
%                                    1 MHz (R = 5^3       =  125)
%                                  200 kHz (R = 5^4       =  625)
%                                  100 kHz (R = 5^4 * 2   = 1250)
%                                   50 kHz (R = 5^4 * 2^2 = 2500)


%% ========================================================== FIR: Target: 25 MHz
%
% Specify a number of FIR lowpass filters for a permutation of:
```

```
% - Start Frequency of the pass band (upper edge)
% - Stop Frequency of the stop band (lower edge)
% - Ripple in pass band
% - Attenuation in stop band
%
%
% See also:
% https://ch.mathworks.com/help/signal/ref/fdesign.lowpass.html
% https://ch.mathworks.com/help/dsp/ref/fdesign.decimator.html


clear all;close all;clc;
% ------------------------------------------------ Input Sampling Frequency in Hz
Fs  = 125e6;

% ------------------------------------------------------------- Decimation Factor
R   = 5;

% -------------------------- Frequency at the Start of the Pass Band; Normalized
% NOTE: The smallest number in Fp must be smaller than the smallest number in
%       Fst (see below).
Fp  = [0.1 0.15 0.2];

% --------------------------- Stop band frequencies ("How steep is the filter?")
% NOTE: The smallest number in Fst must be larger than the largest number in
%       Fp (see above).
Fst = [0.21 0.22];

% ------------------------------------------------------- Ripple in Passband in dB
Ap  = [0.25 0.5 1];

% ------------------------------------------------ Attenuation in Stop Band in dB
Ast = [20 40 60 80];

% ------------------------------------------------------------ Filter Design Objects
Hd  = cell(length(Fp),length(Ap),length(Fst),length(Ast),2);

% ------------------------------ Lengths of Numerators for the Different Filters
% Saves the length of the numerator for each permutation of Fp, Ap, Fst and Ast,
% along with those parameters themselves. Each filter gets a number as well (see
% 't' below).
NumL  = cell(length(Fp),length(Ap),length(Fst),length(Ast),6);

% --------------- Same Thing, for more convenient extraction to file or somesuch
% Saves the length of the numerator for each permutation of Fp, Ap, Fst and Ast,
% but without those parameters.
NumL2 = [];

% Plot as we proceed. This enables color cycling by default.
```

```
figure;hold on;
t = 0;              % total number of filters; filter number
l = 1;              % cell index for Fp
for fp = Fp
    i = 1;          % cell index for Ap
    for ap = Ap
        j = 1;      % cell index for Fst
        for fst = Fst
            k = 1; % cell index for Ast
            for ast = Ast
                d = fdesign.decimator(...
                    R,...
                    'lowpass',...
                    'Fp,Fst,Ap,Ast',...
                    fp,...
                    fst,...
                    ap,...
                    ast);

                Hd{l,i,j,k,1} = design(d,'SystemObject',true);
                Hd{l,i,j,k,2} = t;

                NumL{l,i,j,k,1} = fp;
                NumL{l,i,j,k,2} = ap;
                NumL{l,i,j,k,3} = fst;
                NumL{l,i,j,k,4} = ast;
                NumL{l,i,j,k,5} = t;
                NumL{l,i,j,k,6} = length(Hd{l,i,j,k,1}.Numerator);
                NumL2 = [NumL2 NumL{l,i,j,k,6}];
                scatter(t,NumL{l,i,j,k,6});
                k = k+1;
                t = t+1;
            end
            j = j+1;
        end
        i = i+1;
    end
    l = l+1;
end
% hfvt = fvtool(Hd{:,:,:,:,1},'ShowReference','off','Fs',[Fs]);
```