Chilipepper User's Guide

Toyon Research Corp.
embedded@toyon.com

Table of Contents

1 Foreword 1
2 Introduction 2
3 Interface Signals 3
3.1 MCU and RF-FE Signaling 4
3.2 Transmit Signaling and Digital to Analog Control (DAC) Driver 5
3.3 Receive Signaling and Analog to Digital Control (ADC) Driver 6
3.4 Constraints 7
4 Chilipepper DIP Switch Settings 8
5 FPGA to MCU Serial Protocol 9
6 Reflashing MCU Firmware 10
7 Precautions 12
8 Reference Designs 13
9 References 14

1 Foreword

When I was a senior in college in 1997 I was like any other engineer. I wanted to build something. I wanted to make something work. But, when I went looking for jobs I saw there were two groups. The first did simulations and the other implemented stuff. I wanted to do both. So, like anyone who is unhappy and doesn't really know what to do with their life I went to graduate school.

Entering graduate school I was thrilled with the math and many of the labs were great. But, where were the radios? I thought, ok once I move on from this master's stuff and start graduate work I'll get to build something. But, besides some very minor catastrophic failures with hardware, I mostly published lots and lots of paper.

After graduating in 2003 and entering industry I thought ok this is it. I'm finally going to build something that works! But, I found the same great divide there was in 1997. There were systems engineers and there were hardware engineers. They talked. But, they didn't talk very well. Furthermore, there was almost no way to build a working radio besides building completely custom radio frequency (RF) front ends. And, we began to do this culminating in several custom single and multiple-input multiple-output (MIMO) transceivers. But, boy were they expensive and inflexible.

Chilipepper and the general MATLAB to HDL framework presented is the culmination of over a decades worth of frustration and exploration. For the first time I feel there is a way for a single engineer, or small team, to build a functional wireless radio that has a straightforward path to a real product.

I hope you enjoy Chilipepper. And if you don't let's figure out how to make it better.

-rich cagley Oct. 2012

2 Introduction

Chilipepper is a radio frequency front end (RF-FE) meant for rapid prototyping of physical layer waveforms. The board complies with the VITA 57 standard for FPGA Mezzanine Cards (FMC). The radio itself is built around a Lime Microsystems LMS6002 radio frequency integrated circuit (RF-IC). The LMS6002 is a highly integrated device with most major functional elements within the common package. This includes analog and digital data converters, both transmit and receive synthesizers, filtering, and multiple stages of amplification and gain control.

While the LMS6002 is a highly integrated device, it is still complicated, requiring many weeks of development and some relatively expensive laboratory equipment in order to make the device functional and operate well. As Chilipepper is targeted to those just getting started with wireless radio design, we have placed a microcontroller unit (MCU) onboard to handle calibration and configuration duties. The MCU also masks the register control of the LMS6002, which requires and nondisclosure agreement (NDA) to be in place with Lime Microsystems. This is something most users do not want to deal with during initial development.

3 Interface Signals

In this section we will cover the interface and control signals that you will need to have present in your FPGA design. We suggest mapping MCU and RF-FE signaling to a microcontroller running on the FPGA. The TX/DAC and RX/ADC signals should be driven by FPGA logic.

The RF-FE is driven by a 40 MHz oscillator that is present on Chilipepper. This source drives the synthesizers and other circuitry on the RF-IC. While not mandatory, we suggest basing your baseband design around this clock, which is returned with the schematic signal name in Table 1.

Table 1 - Clocking.

Signal Name	Function
PLL_CLK_OUT	Main clock from RF-RE to the FPGA. This is a 40 MHz signal.

3.1 MCU and RF-FE Signaling

There is an AVR microcontroller present on Chilipepper that is responsible for the bulk of calibration and control of the RF-FE. The relevant signals are given in Table 2. We suggest the following power-on procedure:

1. Disable all clocking and signaling for DAC and ADC lines (all signals in Sections 3.2 and 3.3)
2. Set `TX_EN` and `RX_EN` high (during operation these can be brought low to save power)
3. Bring `MCU_RESET` high then low then high (this resets the MCU)
4. Wait for `MCU_INIT_DONE` to go high

Once the MCU has finished initialization you will also notice that there is a flashing light on Chilipepper indicating the RF-FE is calibrated. You may now turn the power amplifier on/off and change transmit and receive modes.

Table 2 - MCU interface signals.

Signal Name	Function
MCU_UART_RX/TX	Serial interface from FPGA to Chilipepper's AVR MCU. See Section 3 for details.
MCU_INIT_DONE	Signal from MCU to FPGA indicating the MCU has finished calibration. Active high.
MCU_RESET	Signal from FPGA to MCU causing it to reset. Active low.
M1_TR_SW	Transmit/receive switch control for RF-FE. Setting this signal low enables transmit and high enables receive.
M1_PA_EN	RF-FE power amplifier control. Setting this signal low disables the PA and high enables PA. There is no control over power output levels through direct PA interaction and instead this should be done through the serial port interface.
TX/RX_EN	Set these high to enable to TX/RX portions of the RF-FE. The MCU will not begin calibration until these signals are set high

3.2 Transmit Signaling and Digital to Analog Control (DAC) Driver

TX/DAC control is straightforward with signals described in Table 2. A timing diagram is shown in Figure 1. Note that the TX_CLK signal should be half the TX_IQ_SEL signal frequency. Inphase (I) and quadrature (Q) data is interleaved with I corresponding to TX_IQ_SEL being high and Q corresponding to TX_IQ_SEL being low.

Table 3 – TX/DAC interface signals.

Signal Name	Function
TX_CLK	Falling edge latching clock from the FPGA to the DAC. Can be driven up to 40 MHz and should be twice the sample rate of the TX data lines
TX_IQ_SEL	In order to conserve pins, inphase and quadrature signals on transmit are interleaved. This binary signal selects whether I or Q is latched into the DAC during the rising edge of TX_CLK.
TXD	There are 12 bits going to the DAC. TXD0 is the least significant bit and TXD11 is the most significant bit. Data is in 2's complement format.

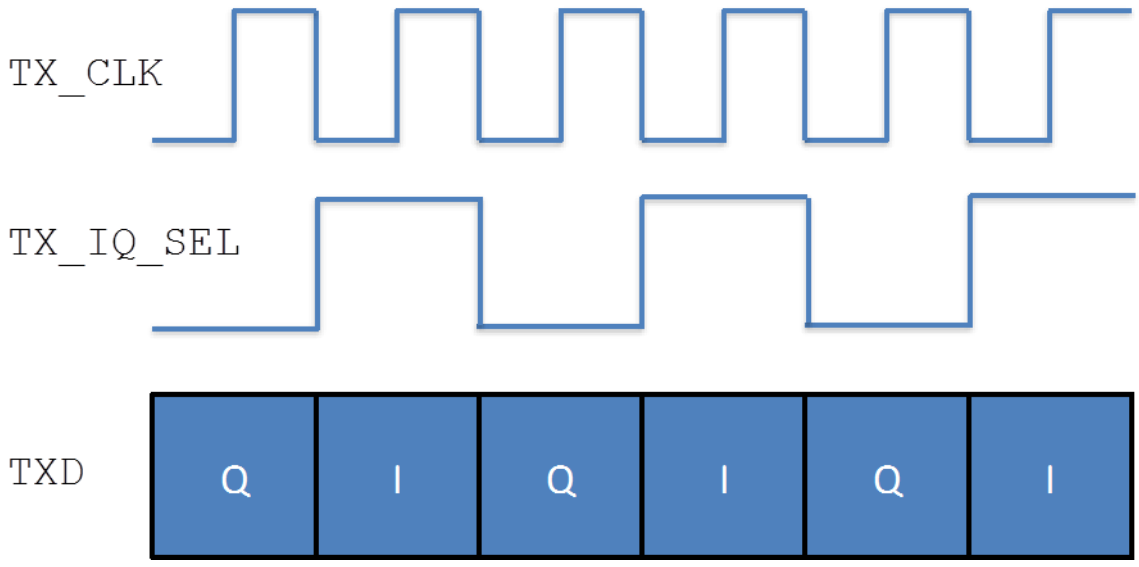

Figure 1 - DAC interface signaling.

3.3 Receive Signaling and Analog to Digital Control (ADC) Driver

RX/ADC signaling is very similar to the

Table 4 – RX/ADC interface signals.

Signal Name	Function
RX_CLK	Rising edge latching clock from the FPGA to the ADC. Can be driven up to 40 MHz and should be twice the sample rate of the RX data lines. Note this clock should not be the same clock that drives your receive processing as you will need to account for skew as shown in Figure 2.
RX_CLK_RET	This clock is from Chilipepper to the FPGA. It can be used to align the returned RX_IQ_SEL and RXD signals with a clock. Currently, this pin is on a non-clocking pin of the FMC connector and as such the user should skew RX_CLK to have a phase offset of 90 degrees earlier compared to the clock used to process receive data.
RX_IQ_SEL	In order to conserve pins, inphase and quadrature signals on receive are interleaved. This binary signal selects whether I or Q is latched into the DAC during the rising edge of RX_CLK. It is sent from Chilipepper to the FPGA.
RXD	There are 12 bits going to the ADC. RXD0 is the least significant bit and RXD11 is the most significant bit. Data is in 2's complement format.

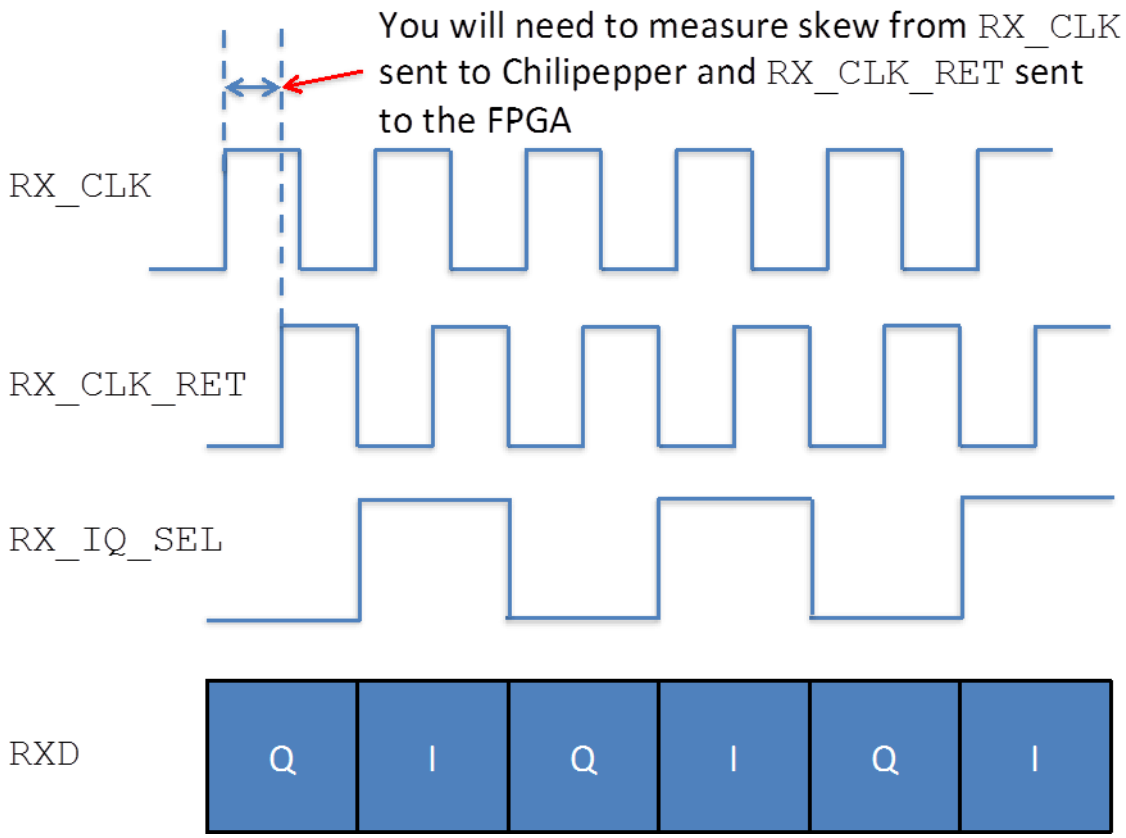

Figure 2 - ADC signaling.

3.4 Constraints

Ideally users will probe signaling from the FPGA to Chilipepper to determine drive strength and any overshoot. On the ZED board with 2.5V signaling we used FAST switching and the lowest possible drive strengths of 4 for clocking. Please refer to Appendix C.

4 Chilipepper DIP Switch Settings

DIP switches on the board control transmit and receive modes. *Note, only mode 1 is currently functional. Please inquire if other modes are desired.

[1]TDD in 2.4 GHz band
[2]FDD in 2.2 GHZ band
[3]Wideband receive only

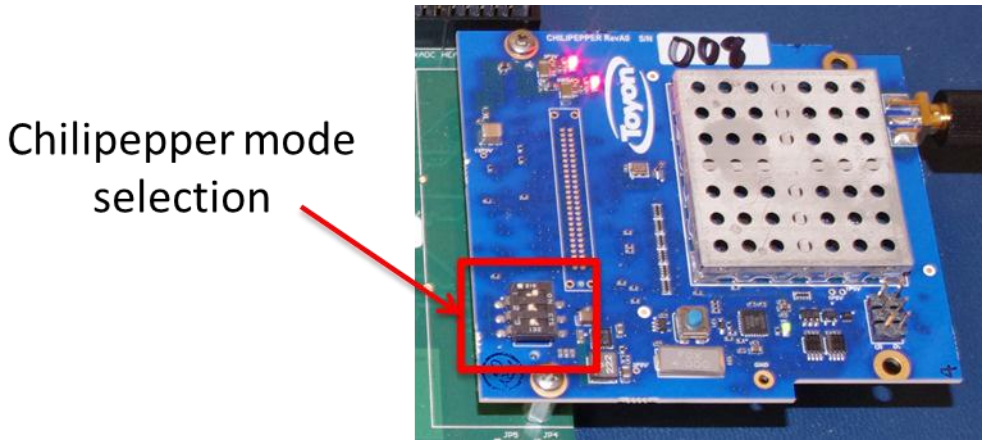

Figure 3 - Illustrating default state of DIP switches.

5 FPGA to MCU Serial Protocol

A simple four byte packet is used to configure and control Chilipepper. The FPGA must follow the format illustrated in Figure 4. Use 9600 baud with no parity checking or hardware flow control.

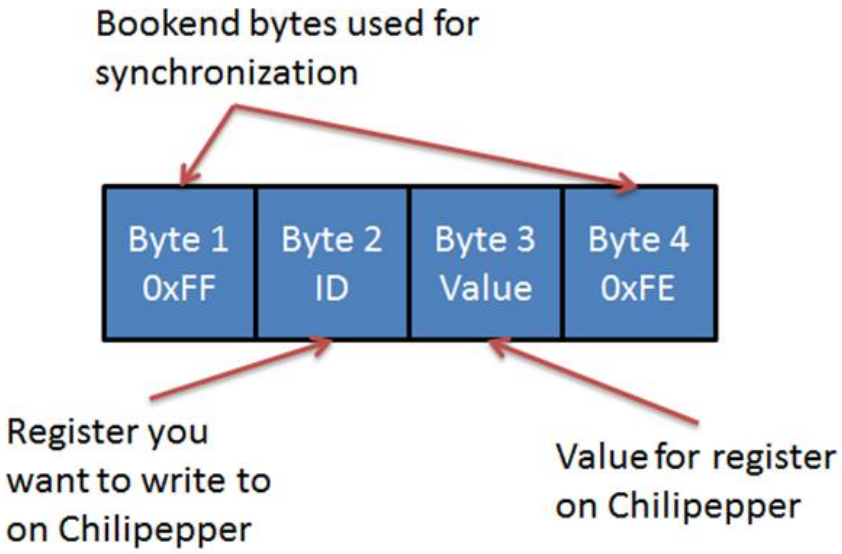

Figure 4 - Serial packet format for Rx/Tx with Chilipepper.

Tables 5 and 6 provide information on control that made available to the FPGA. Note that Chilipepper will not initiate transmit packets back to the FPGA. These will only take place in response to a packet sent from the FPGA to Chilipepper. The FPGA must read the four return bytes every time a packet is sent.

Table 5 - Serial packet transmission from FPGA to Chilipepper.

Register ID	Value range	Functional description
0x00	0 or 1	Controls LED on Chilipepper. 0 is off and 1 is on.
0x01	0 to 25 (default 0)	Controls Tx gain. Value is in dB.
0x02	0 to 57 (default 14)	Controls Rx gain. Value is in dB. Note that values above 30 have resolution of 3. Hence, 30, 31, 32 all generate equivalent Rx gains.

Table 6 – Return serial packet transmission from Chilipepper to FPGA.

Register ID	Value range	Functional description
0x00	0 or 1	Query Chilipepper status. 0 – indicates problem. 1 – everything operating normal.
0xYZ	NA	Any other ID/value combination sent from FPGA is simply returned.

6 Reflashing MCU Firmware

You will need to install Atmel Studio 6.0 along with any service packs. A screenshot of installed tools is shown in Figure 3.

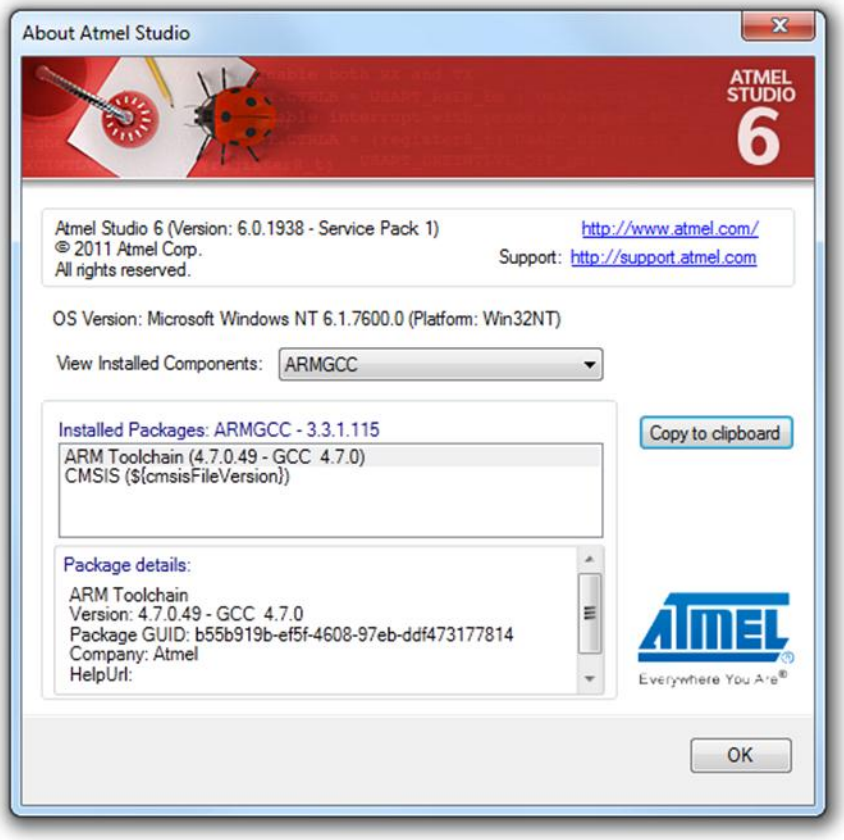

Figure 5 - Current version used in development.

You will also need to install the driver for the programmer and optionally update the firmware.

https://www.olimex.com/Products/AVR/Programmers/AVR-ISP500/

Once your software and drivers are installed, you can use Atmel Studio.

1. Open Atmel Studio 6.0 and the programmer dialog box "Tools -> Device Programming"
2. Select Tool/Device/Interface and then click "Apply"
3. Select "Production file" on the left bar
4. Select the elf file and the "Flash" checkbox and then click "Program" button

Screenshots in Figures 4-5 illustrate the process.

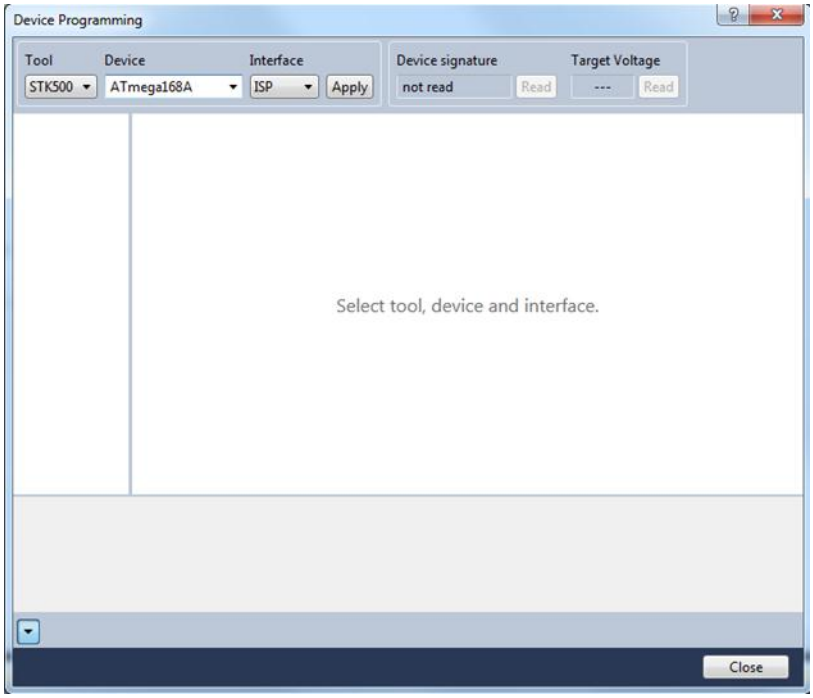

Figure 6 - Atmel Studio Device Programming dialog.

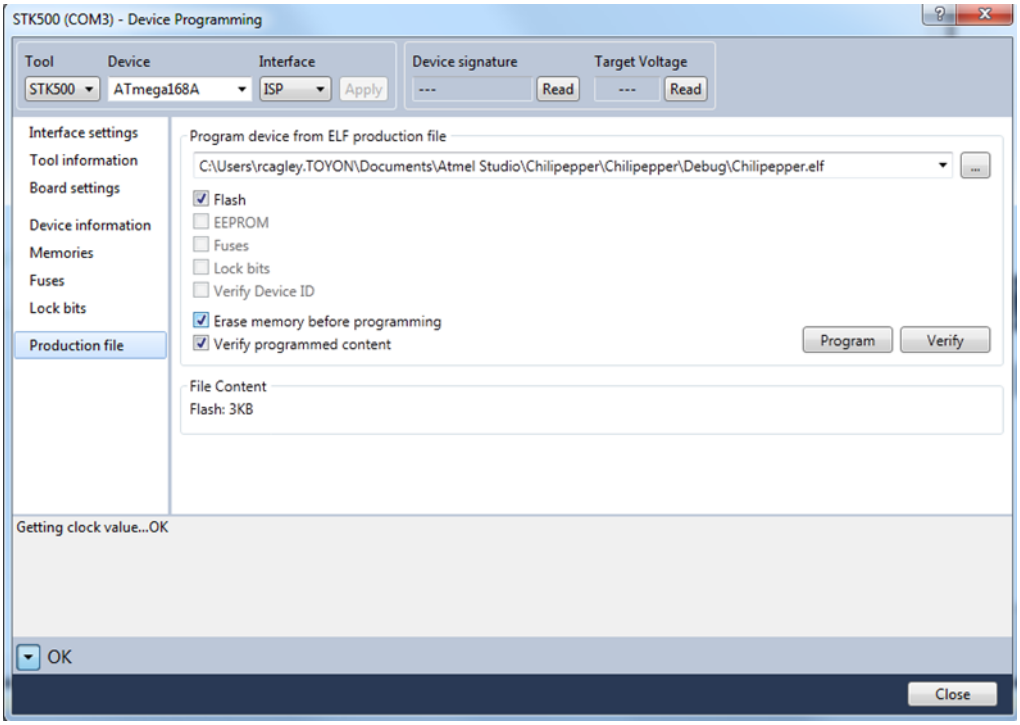

Figure 7 – Production File programming box.

7 Precautions

- Always use proper ESD handling precautions.
- When no antenna connected please attach 50 ohm passive load.

8 Reference Designs

There is Quadrature Phase Shift Keying (QPSK) example design and set of laboratory exercises. Please contact embedded@toyon.com for information. This is all based on MATLAB to HDL code generation using Mathworks tools. Why? They're awesome. Seriously, they're awesome.

9 References

[1] R. E. Cagley, S. A. McNally, B. T. Weals, R. A. Iltis, S. Mirzaei, and R. Kastner, "Implementation of the Alamouti OSTBC to a Distributed Set of Single-Antenna Wireless Nodes", in Proc. IEEE Wireless Communications and Networking Conference, Hong Kong, Mar. 2007.

[2] R. E. Cagley and Nicholas Hogasten, "Bridging the Gap Between Advanced Image Processing and Hardware Design," Synopsys Users Group Conference, June 2010.

R. E. Cagley, "Automating FPGA/ASIC Design Workflow with MATLAB", The MathWorks Signal Processing Virtual Conference, May 2011.

Appendix A FAQ

[Q1] In my signal from the ADC I see a DC offset. How do I remove it?

[A1] The best way is to implement a carrier offset and then apply a high pass or bandpass filter. This can easily be done with a cascade-integrator-comb (CIC) filter. Alternatively if you want to do direct-conversion you can simple track and remove the DC bias, e.g., dc = (1-alpha)*dc + alpha*adc_signal.

[Q2] What does the flashing LED on Chilipepper mean?

[A2] This indicates that either your dip switches are not in a valid mode or that Chilpepper has not been correctly configured. Please check the dip switches and ensure that proper control signaling is followed according to Section 3.

Appendix B What is a Chilipepper?

We name all our boards after fish because several of us fish. Chilipepper is the common name for *Sebastes goodei* from the family *Scorpaenidae*, which is a deep water rockfish found off the California coast. It is normally found over rocky bottoms and feeds on small crustaceans, squid, and fishes. It can live up to 16 years and opposite from most fish, it gives birth to live young. The current record chilipepper is 22 inches long and 5.25lbs. Luckily your Chilipepper is not that big.

Appendix C Example Constraints

```
##############################################################################
# PL clocks and reset
##############################################################################
NET clock_generator_0_pll_pin LOC = D18    | IOSTANDARD = LVCMOS25;
NET clock_generator_0_pll_pin TNM_NET = clock_generator_0_pll;
TIMESPEC TS_clock_generator_0_pll = PERIOD clock_generator_0_pll 40.000 MHz;
##############################################################################

##############################################################################
# Tx - FMC interface at 2.5V
##############################################################################
NET clock_generator_0_tx_clk_pin          LOC = C17   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_tx_iq_sel_pin LOC = B16   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[0]          LOC = A18   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[1]          LOC = A19   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[2]          LOC = E20   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[3]          LOC = G21   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[4]          LOC = F19   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[5]          LOC = G15   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[6]          LOC = E19   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[7]          LOC = G16   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[8]          LOC = G19   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[9]          LOC = A16   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[10]   LOC = A17   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET dac_driver_axiw_0_txd_pin[11]   LOC = C18   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;

##############################################################################

##############################################################################
# Rx
##############################################################################
NET clock_generator_0_rx_clk_pin                LOC = J18   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
# goes out to rx ADC
NET adc_driver_axiw_0_rx_iq_sel_pin       LOC = N19   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[0]                LOC = M21   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[1]                LOC = J21   | IOSTANDARD = LVCMOS25;
```

```
NET adc_driver_axiw_0_rxd_pin[2]                LOC = M22   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[3]                LOC = J22   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[4]                LOC = T16   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[5]                LOC = P20   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[6]                LOC = T17   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[7]                LOC = N17   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[8]                LOC = J20   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[9]                LOC = P21   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[10]         LOC = N18   | IOSTANDARD = LVCMOS25;
NET adc_driver_axiw_0_rxd_pin[11]         LOC = J16   | IOSTANDARD = LVCMOS25;

##############################################################################

##############################################################################
# MCU interface
##############################################################################
#NET axi_uartlite_0_RX_pin                      LOC = L21   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
#NET axi_uartlite_0_TX_pin                      LOC = R19   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_mcu_reset_pin              LOC = K20   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_tx_en_pin                       LOC = D22   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_tr_sw_pin                       LOC = D20   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_rx_en_pin                       LOC = C22   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_pa_en_pin                       LOC = E21   | IOSTANDARD = LVCMOS25 | DRIVE = 4 | SLEW = FAST;
NET mcu_axiw_0_init_done_pin              LOC = K19   | IOSTANDARD = LVCMOS25;
##############################################################################

##############################################################################
# LEDs
##############################################################################
NET clock_generator_0_LOCKED_pin                LOC = T22  | IOSTANDARD=LVCMOS33;  # "LD0"
NET mcu_axiw_0_blinky_mcu_pin                            LOC = T21  | IOSTANDARD=LVCMOS33;  # "LD1"
NET tx_axiw_0_blinky_tx_pin                              LOC = U22  | IOSTANDARD=LVCMOS33;  # "LD2"
NET dac_driver_axiw_0_blinky_dac_driver_pin  LOC = U21  | IOSTANDARD=LVCMOS33;  # "LD3"
NET adc_driver_axiw_0_blinky_adc_driver_pin  LOC = V22  | IOSTANDARD=LVCMOS33;  # "LD4"
NET rx_axiw_0_blinky_rx_driver_pin           LOC = W22  | IOSTANDARD=LVCMOS33;  # "LD5"
NET axi_gpio_0_GPIO_IO_pin                               LOC = U19  | IOSTANDARD=LVCMOS33;  # "LD6"
NET axi_gpio_0_GPIO2_IO_pin                         LOC = U14  | IOSTANDARD=LVCMOS33;  # "LD7"

##############################################################################
```