

DECEL week – tutorial 3: Dedicated low level solution

- 1 Aim: run the B-scan faster using only the Redpitaya board (servomotor control + trig + signal recording => FPGA reprogramming)

In this tutorial, we propose to generate a new bit-stream for the FPGA that handles sequentially the servomotor control, a TTL output signal for triggering the ultrasound pulse and the signal recording for each angle swept by the servomotor.

2 Connections

Now the Redpitaya board generates the TRIG signal for the PULSER board and the PWM signals for controlling the servomotor. So both signals have to be connected accordingly:

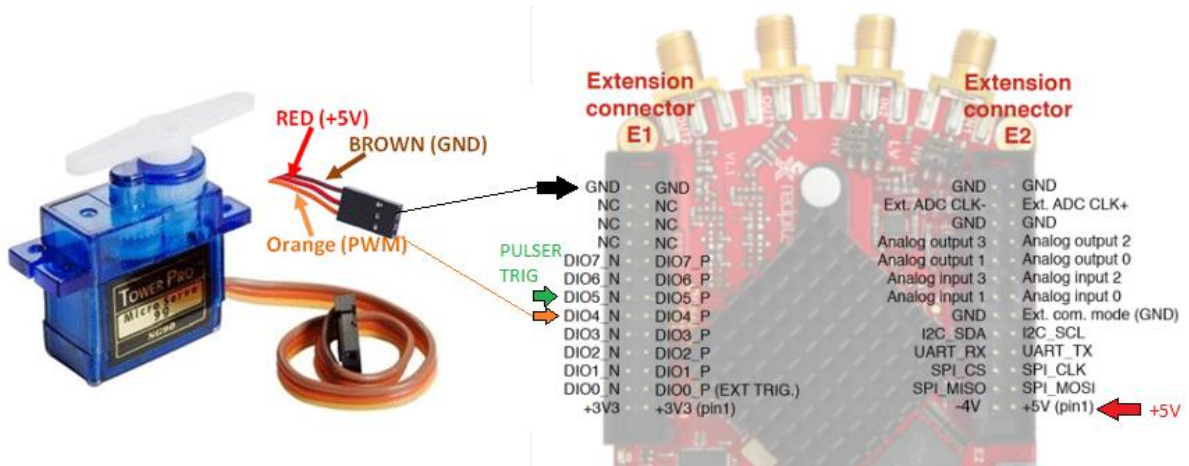


Figure 1 – Connections for servomotor and TRIG signal for the Pulser board

3 FPGA reprogramming

First download the following repository: http://decel.univ-tours.fr/data/DECEL_tuto3_V02.zip (prepared by Prof. Jose Carlos Alves). Unzip it on the D: disk.

In the folder “Rpaquire” open the Rpaquire Vivado project (with Vivado 20,1 installed on the lab’s computers).

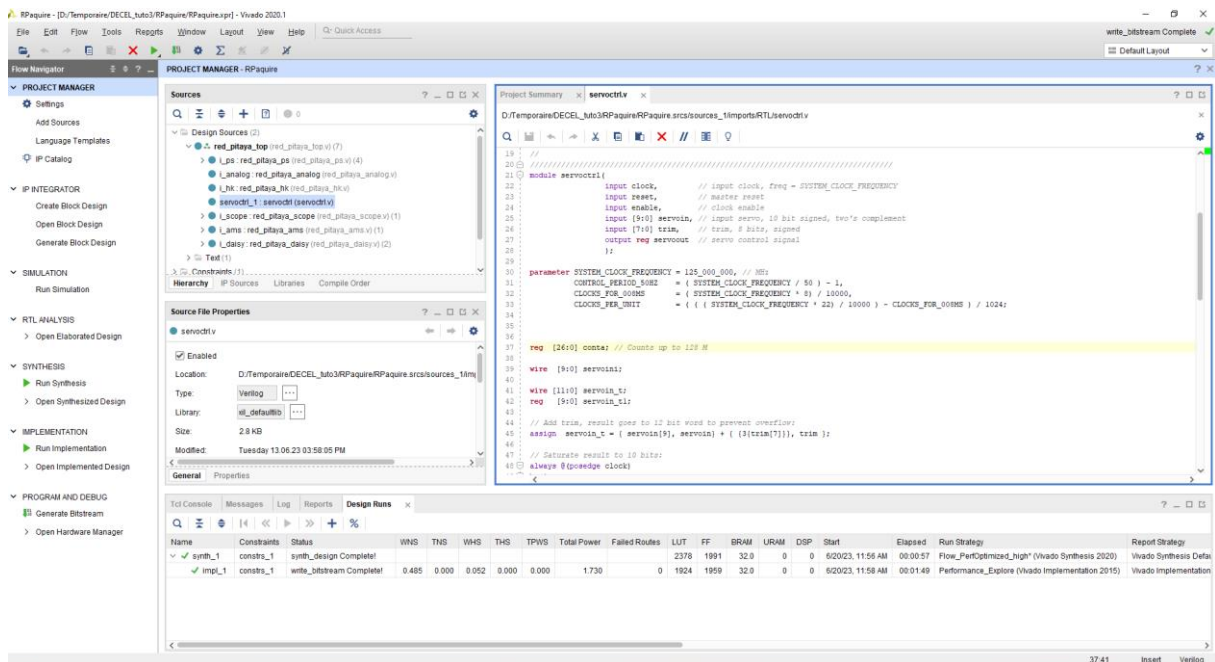




Figure 2 – Vivado project screenshot

Open the structure of the project and Read/Comment the code. You can open the `red_pitaya_scope.v` and change the value of `SYSTEM_HWID` to personalize the value that can be read from register `0x40140000` (using the monitor program, see below).

More explanation on the architecture will be given during the session. For instance, you can see how the servomotor driving is performed.

Then run the synthesis  and generate the bitstream. 

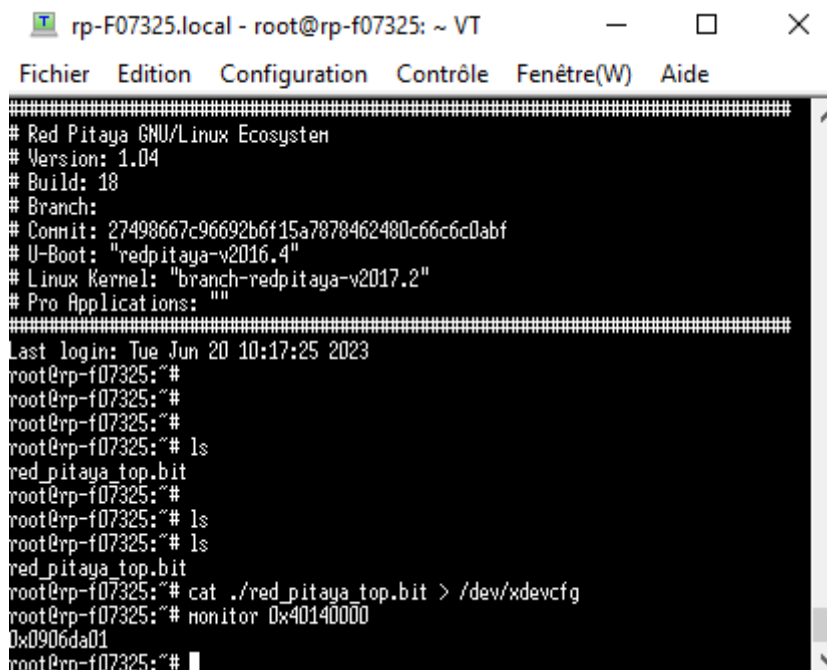
At the end of this process the bitstream ready to be loaded in the FPGA could be found at:

```
...\Rpaquire\RPaquire.runs\impl_1\red_pitaya_top.bit
```

Then open Teraterm (ssh terminal like Putty) and open ssh connection with the following parameters:

```
IP : rp-F0XXXX.local
login : root
pass : DEEF0XXXX
```

(NB : XXXX correspond to the end of your redpitaya MAC address).



```
rp-F07325.local - root@rp-f07325: ~ VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
#####
# Red Pitaya GNU/Linux Ecosystem
# Version: 1.04
# Build: 18
# Branch:
# Commit: 27498667c96692b6f15a7878462480c66c6c0abf
# U-Boot: "redpitaya-v2016.4"
# Linux Kernel: "branch-redpitaya-v2017.2"
# Pro Applications: ""
#####
Last login: Tue Jun 20 10:17:25 2023
root@rp-f07325:~#
root@rp-f07325:~#
root@rp-f07325:~#
root@rp-f07325:~# ls
red_pitaya_top.bit
root@rp-f07325:~#
root@rp-f07325:~# ls
red_pitaya_top.bit
root@rp-f07325:~# ls
red_pitaya_top.bit
root@rp-f07325:~# cat ./red_pitaya_top.bit > /dev/xdevcfg
root@rp-f07325:~# monitor 0x40140000
0x0906da01
root@rp-f07325:~#
```

Figure 3 – Teraterm. SSH connection to Redpitaya for bitstream loading into FPGA.

They drag and drop the generated bitstream in the console. Then enter `> ls`. If the operation is successful you should see the `red_pitaya_top.bit` listed (as shown in the previous image).

In order to load the bitstream into the FPGA you enter:

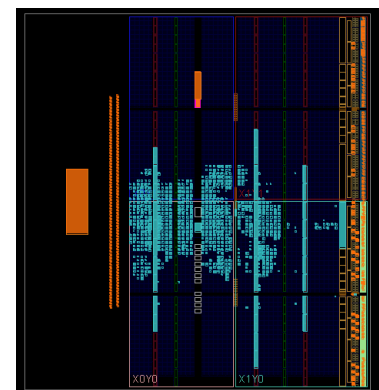
```
> cat ./red_pitaya_top.bit > /dev/xdevcfg
```

To check that this configuration is effective in the FPGA, enter:

```
> monitor 0x40140000
```

Reading the address `0x40140000` should return `0x0906_DA01`

(or the value you have configured for `SYSTEM_HWID`).



Register map implemented:

Figure 4 – Implemented design

The default values after FPGA configuration are zero, unless a different value is refi

Servo motor controller (output is pin DIO_4N)

Address	Function	Read, Write ¹	Bits ²	Description
0x4010 0020	Enable/disable servo controller	W/O	[0:0]	1: enable generation of servo control signal 0: set servo control signal to zero
0x4010 0024	Servo trim	W/O	[7:0]	8-bit signed offset added to the servo input data
0x4010 0028	Servo position	W/O	[9:0]	10-bit signed position, zero is the middle position of the servo -512 generates a 0.7 ms high pulse, +511 generates a 2.3 ms high pulse

Output trigger generation (output trigger is pin DIO_5N)

Address	Function	Read, Write ¹	Bits ²	Description
0x4010 003C	Output trigger enable	W/O	[1:0]	0X: disable trigger generation, set output trigger to zero 10: enable software generated trigger (trigger is data written to 0x40100030) 11: enable hardware generated trigger
0x4010 0030	Software generated trigger	W/O	[0:0]	If trigger enable is 10z, the LSB of data written to this register is applied to the trigger output (pin DIO_5N)
0x4010 0034	Hardware trigger period	W/O	[31:0]	Period of the hardware trigger signal, in number of cycles of the 125 MHz clock. Default value is 12 500 000 for generating a 100 ms period (10 Hz)
0x4010 0020	Hardware trigger high-time	W/O	[31:0]	Duration of the high-time of the trigger pulse, in number of cycles of the 125 MHz clock. Default value is 125 000 for a high-time equal to 1 ms

Data acquisition

Address	Function	Read, Write ¹	Bits ²	Description
0x4010 0014	Decimation factor	RW	[15:0]	Set the decimation factor to N, defaults to 16 (see note ³)
0x4010 0018	Decimation average mode	RW	[0:0]	If set to 1, the output sample is the average of the previous N samples when N=1, 2, 4, 8 or 16, where N is the decimation factor; if set to 0 the output sample is the single sample acquired at each N samples (see note ³)
0x4011 xxxx	Data RAM buffer channel A	R/O	[31:0]	Reading from address 0x4011 0000 + (4*addr), addr = 0..16383, reads the 32k 16-bit signed samples acquired from channel A (sample k in the low 16 bits, sample (k+1) in the high 16 bits).
0x4012 xxxx	Data RAM buffer channel B	R/O	[31:0]	Reading from address 0x4012 0000 + (4*addr), addr = 0..16383, reads the 32k 16-bit signed samples acquired from channel B (sample k in the low 16 bits, sample (k+1) in the high 16 bits). See note ⁴ .
0x4010 0004	Acquisition is idle	R/O	[0:0]	Status of the acquisition process: 1=acquisition is idle waiting for trigger; 0=acquisition is running.
0x4010 0010	Arm signal acquisition	W/O	[0:0]	Writing 1 to this register arms the signal acquisition process and the signal acquisition will start with the next trigger event, either a high pulse in pin DIO_5P (input hardware trigger) or a software generated trigger (register 0x4010 0008)

0x4010 0008	Start signal acquisition	W/O	[0:0]	Writing 1 to this register starts the signal acquisition process. This signal is OR'ed with the external hardware trigger input connected to pin the extension connector DIO_5P. To start the acquisition, the process must be armed by writing 1 to register 0x4010 0010 (arm acquisition)
-------------	--------------------------	-----	-------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Miscellaneous

Address	Function	Read, Write ¹	Bits ²	Description
0x4010 0004	Reset clock cycle counter	W/O	[-]	Writing to this address sets to zero the 64-bit clock cycle count running at 125 MHz (the data written is not relevant)
0x4010 0008	Clock cycle count, low 32-bit word	R/O	[31:0]	Reads the low 32 bits of the clock counter and stores the high 32-bit for a subsequent read of the high 32-bits (address 0x4010 000C). This register wraps around at approximately each 34.3597 seconds.
0x4010 000C	Clock cycle count, high 32-bit word	R/O	[31:0]	Reads the high 32 bits of the clock cycle count. To read a coherent 64-bit value, this register must be read only after performing a read of the low 32-bits (register 0x4010 0008). This register is incremented at approximately each 34.3597 seconds.
0x4014 0000	Read the hardware ID register	R/O	[31:0]	Returns the 32-bit value set in the hardware system with parameter SYS-TEM_HWID (see Verilog file red_pitaya_scope.v)

Notes:

¹RW: read/write register, RO: Read only register, WO: Write only register

²Only the bits referred are used by the logic circuit; signed data shorter than 32 bits is just truncated to the least N significant bits indicated.

³The decimation factor sets the sampling frequency by dividing the master 125 MHz clock by the value loaded into this register (defaults to 16, Fs=7.8125 MHz). This can be any integer from 1 to $2^{17}-1=262143$ (sampling frequencies from 125 MHz to 476.8 Hz) but if the averaging mode is active (register 0x40100018 – Average decimation - set to one) this value should be only set to the following values: 1: Fs=125 MHz, 2: Fs=62.5 MHz, 4: Fs=31.25 MHz, 8: Fs=15.625 MHz and 16: Fs=7.8125 MHz (defaults to 16, Fs=7.8125 MHz).

⁴The data buffers should be accessed only when the acquisition process is not running, otherwise the signal retrieved may include samples from previous acquisitions. The acquisition process is inactive when reading 1 from register 0x4010 0004 (Acquisition is idle).

4 Low level acquisition program

The FPGA is now configured but data should be passed from the FPGA to the processor and stored at the OS level. To do so a C program have been developed.

Open the file `Rpaquire_SW\src\acq\acq.c` with Notepad++ and read this code. You might want to modify this program later (ex: implementation of software threshold detection, etc.).

In order to transfer this program to the Redpitaya, download the zip folder of this version of the program here (version adapted by Rémi Busseuil): <http://decel.univ-tours.fr/data/decel.zip>

Drag and drop the file in Teraterm and unzip the folder on the redpitaya by typing:

```
> unzip decel.zip
```

Using `cd` command go into `DECEL/SW` and then compile the program:

```
> make
```

The program should compile without issue. Then go into `DECEL/bin` and execute the program:

```
> ./acq
```

You should have one acquisition done. You can download the acquisition file with Teraterm using SCP File→SSH-SCP as illustrated in Figures 5 and 6, you need to enter the path of your acquisition (ex: from `/root/DECEL/bin/dataout00.bin` to the Downloads folder of Windows as illustrated).

```

rp-f07325.local - root@rp-f07325: ~/DECEL/...
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
root@rp-f07325:~/DECEL# cd SW
root@rp-f07325:~/DECEL/SW# ls
acq.c CONST.h Makefile read_write.c read_write.o setup.h
acq.o main.c pwm.c read_write.h setup.c setup.o
root@rp-f07325:~/DECEL/SW# make
gcc acq.o setup.o read_write.o -o ../bin/acq
root@rp-f07325:~/DECEL/SW# ls
acq.c CONST.h Makefile read_write.c read_write.o setup.h
acq.o main.c pwm.c read_write.h setup.c setup.o
root@rp-f07325:~/DECEL/SW# cd ..
root@rp-f07325:~/DECEL# cd bin
root@rp-f07325:~/DECEL/bin# ls
acq dataout00.bin
root@rp-f07325:~/DECEL/bin# ./acq
FPGA Hardware ID = 0906_DAO1
Start acquiring...
Nloops=59
Clocks acquire data: 561745 (4.493960 ms)
Clocks read data: 453926 (3.631408 ms)
-----
Total: 8.125368 ms
Clocks save data: 213242 (1.705936 ms)
Done
root@rp-f07325:~/DECEL/bin#
    
```

Figure 5 - Teraterm. SSH connection to Redpitaya for acquisition.

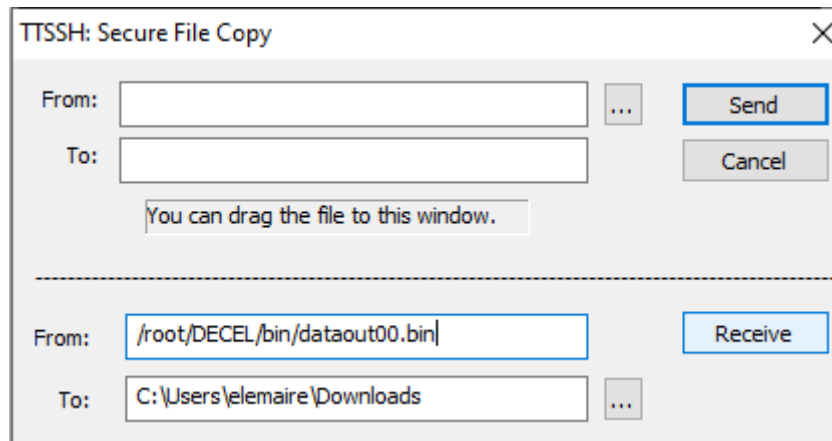


Figure 6 - Teraterm. SCP menu for file transfert from Redpitaya to computer.

Then you can plot the content of the acquisition file with Matlab using the program given in the repository (RPaquire_SW/MATLAB/plotdata.m: reads binary data file dataout.bin created with the acq program and plots the signal acquired).

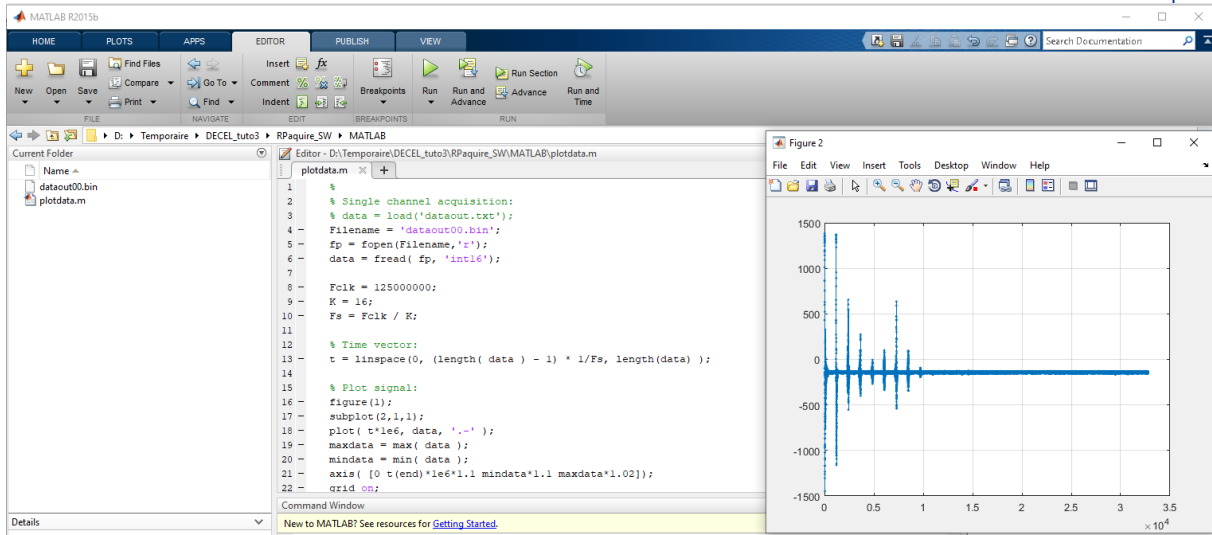


Figure 7 – Matlab. Data loading and plot.

Everything is now ready for a B-scan, just read carefully `acq.c` in order to find how to do the servomotor scan and to collect all the RF lines for each angles. Finally adapt your Matlab program accordingly in order to plot an image (use `mesh()`).

The scan is now faster than before and you can think about averaging, post-treatment and image reconstruction. In the following figure, the scan is done/plot twice (the probe goes from the minimum angle to the max angle and comes back). Interpret the following figure.

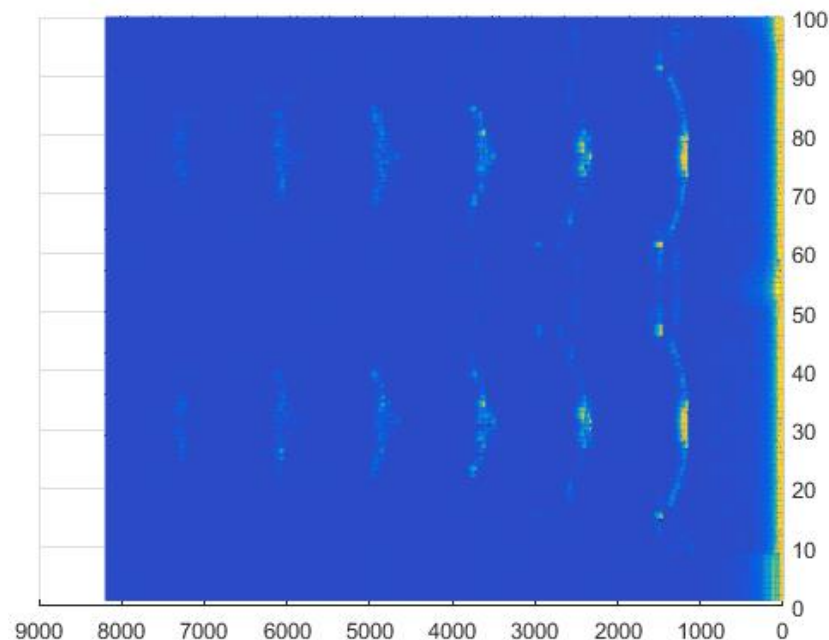


Figure 8 – example of double ultrasound scan with the water tank filled without additional object into it.