Redpitaya: fourth example

G. Goavec-Mérou, J.-M Friedt

February 19, 2020

This tutorial is a sequel to the second one on which it is based. In addition to copying the ADC inputs to the DAC outputs, we wish (Fig. 1) to **mix** the stream of data coming from the ADC with a local numerically controlled oscillator (**NCO**) (Fig. 2).

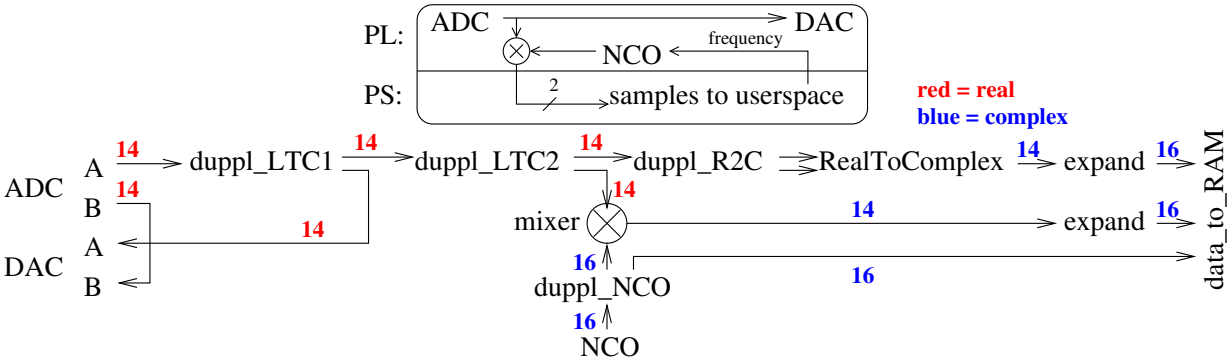

Figure 1: Top: general objective of the frequency transposition of the input signal, and bottom the detailed list of blocks and data path width along the signal processing chain. Colored numbers indicate datawidth, with red being real-valued streams and blue being complex-valued streams.

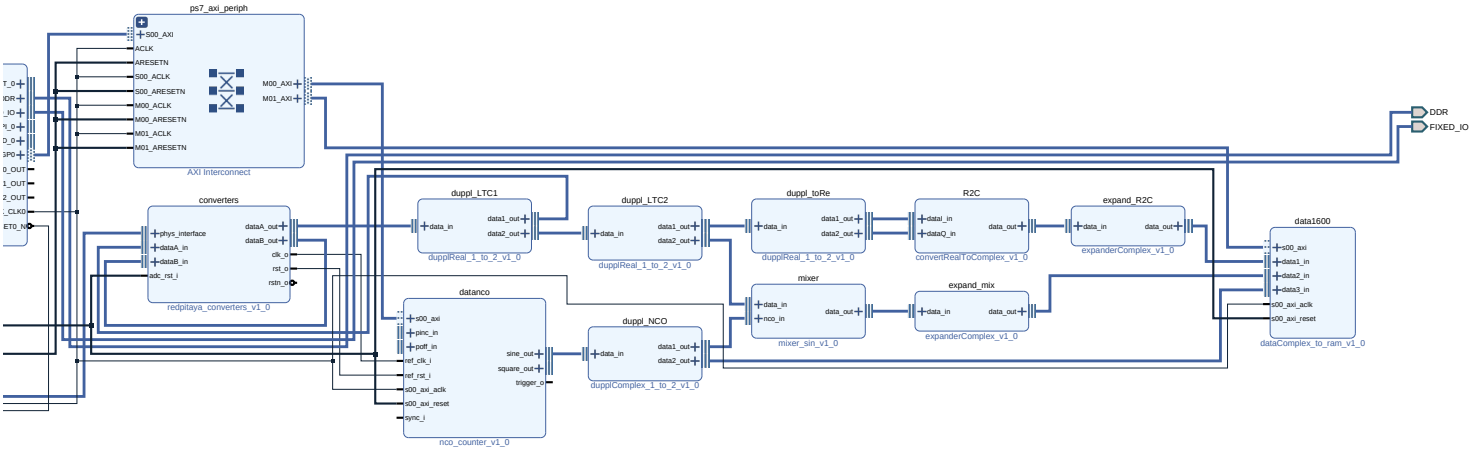

Figure 2: Schematic of the objective and final processing chain described in this document

1 NCO configuration

The ADC to DAC and ADC to RAM blocks are the same as before, with stream dupplication every time a new datastream must reach two users. The novely is now the use of a NCO and mixer. The NCO accepts as a 32-bit (counter size) argument the normalized frequency, and outputs a 16-bit counter value. The new challenge lies in handling varying data sizes: the ADC outputs 14-bit data readily accepted by the DAC and the mixer input (whose input data size is defined as 14-bit wide while the NCO port is set to 16-bit width) but the Data to RAM block can only handle a single data size on all input ports. The option is either to degrade the 16-bit NCO to 14-bit, or to expand the 14-bit ADC and mixer outputs to 16-bits. The latter option was selected to introduce the "expanderComplex" (its application to real valued streams named "expanderReal" is also available) whose function is to add missing Most Significant bits to the input stream to match the expected output stream data size.

Since we feed the Data to Ram block with 3 streams of complex data (real and imaginary) each encoded as 16-bit values, we must read 12-bytes for each new dataset. Hence, in the userspace application, the dataset transfered from the FPGA to the RAM is 12 times the RAM size which is set to 4096 samples.

2 PS: Linux kernel driver

The same driver for communicating with the PL will be requested as before (`data_to_ram`), in addition to which we wish to communicate with the NCO to define the oscillator frequency (`nco_counter` module). This time, the `module_generator` XML configuration file should look like

```
<?xml version="1.0" encoding="utf-8"?>
<project name="tutorial5" version="1.0">
    <ips>
        <ip name ="dataComplex_to_ram" >
            <instance name="data1600" id = "0"
                base_addr="0x43C10000" addr_size="0xffff" />
        </ip>
        <ip name ="nco_counter">
            <instance name="datanco" id = "0"
                base_addr="0x43C00000" addr_size="0xffff" />
        </ip>
    </ips>
</project>
```

The name of the appropriate module has been found by looking for a directory with the same name than the IP in the `linux_driver` directory, while the addresses have been obtained by reading the appropriate fields in Vivado (Fig. 3).

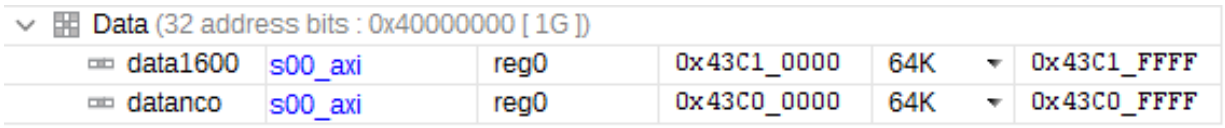

Figure 3: Address range used by the IPs as defined by Vivado.

3 On the Redpitaya ...

For communicating with the NCO to configure its frequency, we will benefit from an existing library function as implemented in `liboscimp_fpga`. The NCO configuration function is defined in "nco_conf.h" and requires as argument the `/dev` entry as defined in the module generator configuration file, the sampling rate f_{ck}, the NCO rate f_o, and the accumulator size acc since as in a DDS, the configuration word is $f_o/f_{ck} \times 2^{acc}$.

```
#include <stdio.h>
#include <stdint.h>
#include <fcntl.h>
#include <unistd.h>
#include "nco_conf.h"

// 3 complex short = 6*2*2 bytes/measurement
#define channels 12

int main()
{char c[4096*channels];
 int fi,fo;
 fi=open("/dev/data1600",O_RDWR);
 fo=open("/tmp/data.bin",O_WRONLY|O_CREAT,0666);
 nco_counter_send_conf("/dev/datanco", 125000000, 10000000, 32, 0, 1, 1);
                        // /dev f_ck=125M f_o=10M acc offs pinc poff
 read(fi,c,4096*channels);
 write(fo,c,4096*channels);
 close(fi);
 close(fo);
}
```

The arguments to the `nco_counter_send_conf()` functions are

1. input frequency f_{in} (125000000 Hz)
2. output frequency $_{out}$ (Hz)
3. accumulator size acc (32)
4. offset $offs$ in the lookup table RAM inducing a constant phase offset (0)
5. phase increment source $pinc$ to specify whether the accumulator increment is driven by an FPGA input (0) or PS input (1)
6. phase offset configuration source $poff$, 0 to drive from FPGA or 1 to drive from PS.

The output frequency of the NCO is then, based on these definitions

$$f_{out} = f_{in} \times \frac{word}{2^{acc}}$$

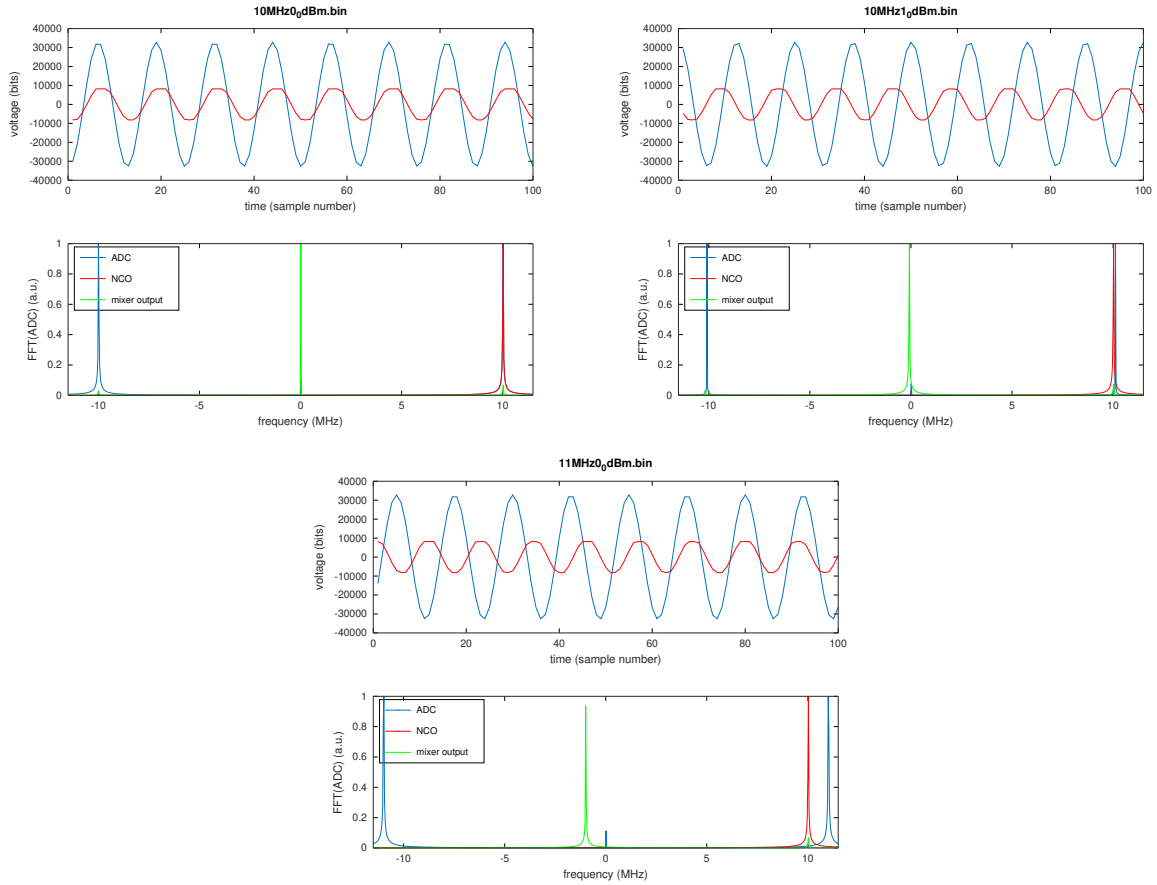

Figure 4: Left to right: 10.0 MHz input signal, 10.1 MHz input signal, and 11.0 MHz input signal. Each chart exhibits (top) the time domain acquisition (ADC) and local oscillator (NCO) real part, and (bottom) the frequency domain Fourier transform of the ADC, NCO and mixer output.