EEC0055 - Digital Systems Design

2019/2020

Laboratory 3
17 November – 24 December 2019

Ultrasonic wind speed and direction sensor

Revision history

date	notes	author
Nov 18, 2019	Preliminary version V0.1	jca@fe.up.pt

1 – Introduction

In this project we will implement an ultrasonic wind speed and direction sensor, based on the variation of the sound speed due to the movement of the air (the sound wave propagation medium) induced by the wind. The system will be designed using the XILINX ISE design environment and the Verilog Hardware Description Language, and implemented in the ATLYS board. Some modules will be provided as the source RTL code, as well as reference implementations as Matlab scripts. The CORDIC module developed in the first project will also be a fundamental building block. The overall design goal is minimize the logic complexity (area) while using a low clock frequency (2MHz) to save power.

2 – Principle of operation

The ultrasonic wind sensor calculates the wind speed and the wind direction by measuring the phase differences between the acoustic signals received by four orthogonally placed receivers. Figure 1 shows the physical arrangement of the ultrasonic transducers. A central transducer continuously transmits a single tone at frequency **Ftx** that is received by the four receivers. In the absence of wind, and assuming the 4 receivers are at the same distance **D** from the transmitter, the four signals received will be in phase (although out of phase from the transmitted signal).

If the air is moving along the direction formed by two opposite receivers (**Rx1** and **Rx3** or **Rx2** and **Rx4**) the sound propagation speed from the transmitter to the receiver downwind will increase and the speed to the upwind receiver will decrease. This will create a phase shift in the signals received, positive in the downwind receiver (signal arrives first) and negative in the upwind receiver. Measuring the phase difference between the signals arriving to two opposite receivers (**Rx1** and **Rx3** or **Rx2** and **Rx4**) will allow to determine the Y and X components of the wind speed along the directions defined by the sensors. Then, using a CORDIC module these values are converted from the rectangular X and Y components of the wind speed to polar coordinates, representing the wind speed modulus and the wind direction.

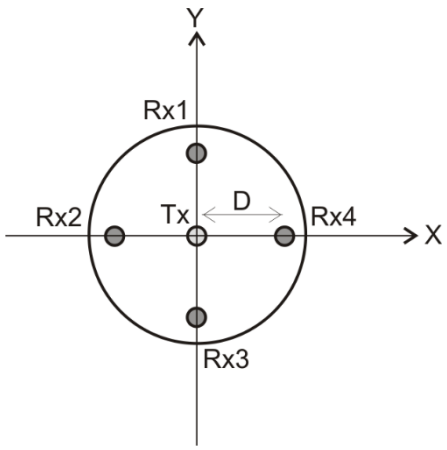

Figure 1 – Physical arrangement of the transmitter and receiver ultrasonic transducers.

The relationship between the phase difference and the wind speed depends on the propagation speed of the sound, which varies approximately linearly with the air temperature. However, because the air temperature will affect equally the X and Y components of the wind speed, the temperature must be considered only for the calculation of the modulus of the wind speed and will not impact the result of the wind direction. The modulus of the wind speed is compensated for the current air temperature by a custom correction module.

Two final calibration modules, based on the linear interpolation over a piece-wise linear curve, adjust the final wind speed and wind direction against deviations due to misalignments of the ultrasonic transducers. The output interface will be based on a low speed bidirectional serial interface (I2C, SPI or UART, to be defined later). The serial interface will also be used to configure various system parameters and to define the calibration functions.

3 – System architecture

Figure 2 presents a simplified functional block diagram of the system. System blocks are:

- The ultrasonic transmitter placed at the center of the sensor is driven by a fixed frequency square wave generated by a clock divider (block **txdriver**). The frequency of this signal will be around 20 kHz but the precise value will be defined later.
- The four acoustic signals acquired by the ultrasonic transducers are amplified and filtered by an external analog circuit, and digitized by a 4-channel multiplexed ADC (as the AD7091-R4 from Analog Devices).
- The block **rxreceiver** implements the serial interface with the ADC and provides the 4 streams of data with the 12-bit digital samples acquired from the 4 ultrasonic receivers.
- To determine the difference of phase between the signals received at opposite receivers, the digital signals coming from the acoustic receivers are converted to their complex representation. To achieve this, a set four modules based on Hilbert FIR filters (blocks **real2cpx**) calculate the real and imaginary components of each signal.
- The instantaneous phase of each signal is calculated from the real and imaginary components by a CORDIC module (the four modules **phasecalc**) and the phase differences between each two opposite receivers are calculated, wrapped to the interval [-180º, +180º] and converted to time by the two modules **phasediff**.
- Then, the phase differences are averaged and converted to the wind speed components along the X and Y axis (modules **phase2speed**).
- An additional CORDIC module converts the X and Y wind speed components to the wind direction and the modulus of the wind speed (module **windrec2pol**).
- As the X and Y wind speed calculation considers a fixed 20 ºC air temperature, the modulus of the wind speed has to be compensated to the current air temperature by module **tempcomp**.
- To compensate misalignments of the acoustic receivers, two calibration modules (**calibwspd**, **calibwdir**) adjust the values of the wind angle and the wind speed implementing linear interpolations over a piecewise linear functional stored in a memory.
- Finally, the serial interface (**serialinterface**) implements the digital interface with the external system.

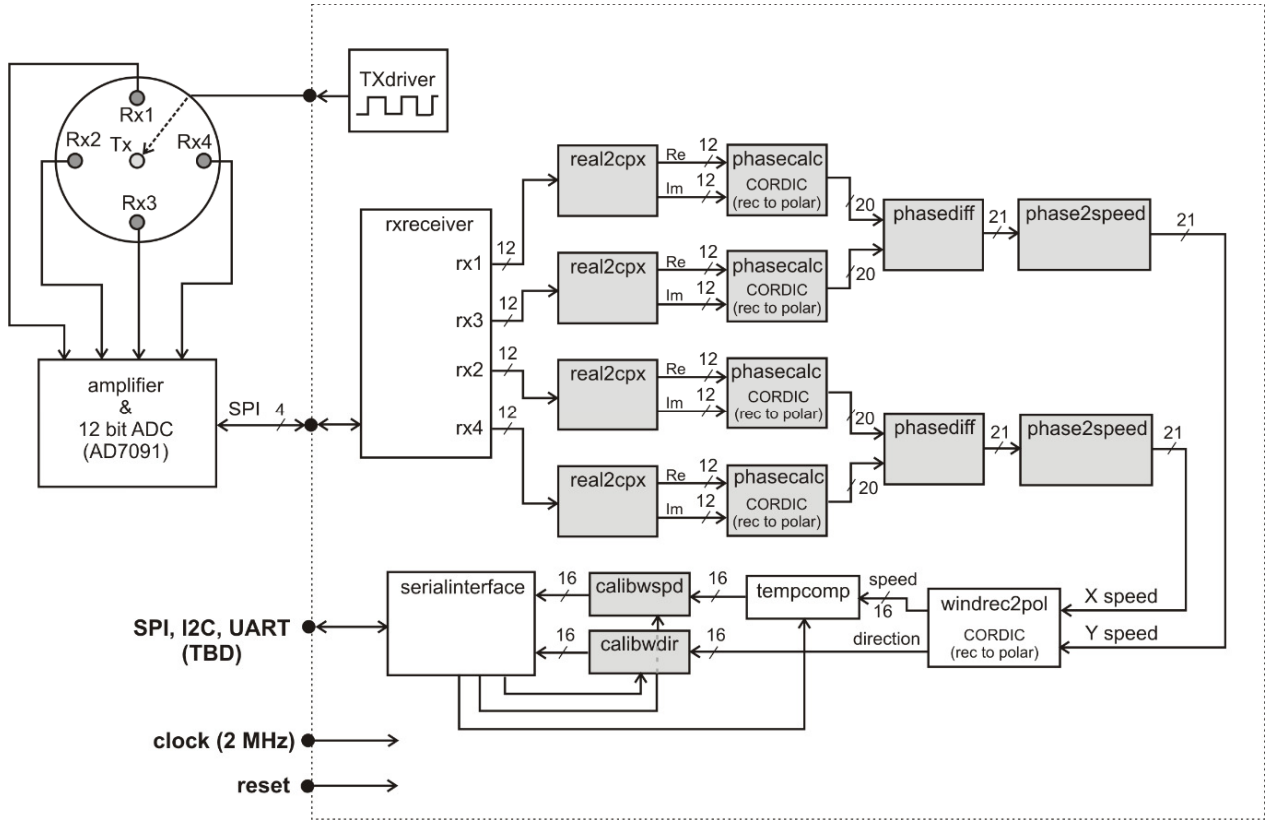

Figure 2 – Wind sensor bock diagram. Note this is only a functional view as some blocks may be shared between the two identical signal processing paths.

4 – Matlab functional model
A Matlab functional model is provided in file 'wind.m'. Download this script to your Matlab working directory and execute 'help wind'. This script simulates the signal processing path for two opposite receivers, from the outputs of module **rxreceiver** (**rx1** and **rx3** or **rx2** and **rx4**) to the output of module **phase2speed**. This script is parameterized with the initial number of bits at various points in the datapath, but the final bit widths will be defined later.

5 – Design implementation
The system must be implemented as a clock synchronous digital system using a single clock of 2 MHz. The sampling rate of the acquired signals is 100 KHz and the same sampling rate must be used until the input of the module **phase2speed** (this means there are only 20 clock cycles to process one signal sample in each channel). After this module the sampling rate is reduced by a factor of 32 (3.125 kHz or 640 clock cycles between samples). The final output values for the wind speed and direction should be made available through the serial interface at the same sampling rate. The latency of the signal processing system can be as high as 10 ms or 20 k clock cycles.

The number of bits presently defined for each point in the datapath is represented in figure 2. However, besides the 12-bit referred for the digital acoustic signals, the bit width and data format of all the other signals along the datapath will be revised soon.

Your role in this project is to design and validate the following key modules (shaded blocks in figure 2):

1) the datapath to calculate the phase difference between the signals acquired by opposite receivers and the conversion to wind speed along each axis;

2) the **windrec2pol** module to convert the X and Y wind speed components to the wind speed modulus and wind direction (this implements the CORDIC algorithm and is similar to the module to use in the phase calculator datapath);

3) the calibration modules (note that the same module will be used for wind speed and wind direction, although using different calibration tables).

The other modules will be provided either as their Verilog source code or as "black boxes" (i.e. only a compiled version will be available).

4.1 – Calibration modules
The calibration modules translate an M-bit input value to an N-bit output value, using a look-up table that represents a piece-wise linear function defined by 16 points and performing linear interpolation to calculate the values between the points stored in the table. The implementation must be parameterized with M and N and the values of these parameters should be between 10 and 20.

From the point of view of this module, the look-up table is accessed as a ROM (read-only memory) with 16 memory locations pre-loaded with the X and Y values that define the piece-wise linear calibration function. This memory may also contain any other data that can help simplify the logic complexity of the module. The circuit to design must be synchronous with the 2 MHz master clock and one calculation may take up to 64 clock cycles. The interface of the module should be:

```
module calibration
   #( parameter M=16, // default value for M
      parameter N=16) // default value for N
    (    input  clock,
         input  reset,
         input  start, // set to 1 to start a new conversion
         output ready, // set to 1 when ready
         input  signed [N-1 : 0 ] X,
         output signed [M-1 : 0 ] Y
    );
```

```
// The lookup table, 16 locations, X and Y pairs:
reg [ N + M - 1 : 0 ] LUTcalib[0:15];

// Load initial contents to the LUT from file "datafile.hex":
initial
begin
  $readmemh( "datafile.hex", LUTcalib );
End


// your code here ...


endmodule
```

… to be continued …