Embedded Systems
Final examination. June 13, 2022

Time limit: 120 minutes.

1 Fixed point arithmetic, ADC and serial port with C (2 points)

In order to implement some kind of filter, we need to compute $Y = a * X$. For instance, if $a = 0.125$ and $X = -1.1$ then $Y = -0.1375$, or if $a = 0.37$ and $X = 2$ then $Y = 0.74$. The parameter a can take any value between $[-0.4, 0.4]$. We would like to do this computation with the ATmega328P AVR microcontroller. So, the voltage X, a frequency modulated zero mean sinusoidal signal [1] of peak amplitude less than 2 V, has been raised 2.5 V before being sampled by the ADC, with reference voltage equal to 5 V. We only take 8 bits of the sampled value, that we will call x.

We want to sample x at $f_s = 20\,\mathrm{kHz}$ and send each computation, after some transformations, through the serial port coded with just one byte. Of course, the one byte computed value, that we will call y, can not always be equal to the desired value Y. The absolute error will be $e = \|Y - y\|$.

1. Explain how to do all the previously described work with the AVR. In this first question there is no restriction in the type of variable you can use, so you can use floating point arithmetic. Write in `C` the computation of y from a and x; Note that even if you use floating point arithmetic to do the computations, the value that you send through the serial port must be coded as a fixed point with B,F=8,F.

2. Consider the examples (a and x given at the beginning of the exercise to compute all the values that appear in the `C` code. This includes, among others, the value x sampled by the ADC and the fixed point 8-bit value y. In particular, report the absolute error e.

3. Repeat the previous questions considering the use of fixed point arithmetic in all computations. Note that there is no restriction in computations made during the compilation.

2 FPGA and VHDL: synchronizing clock enable with data (2 points)

A signal called `clk_en_in` is high the first rising edge of `clk` after `data_in` is ready and is low the next rising edge. We want to design an entity in order to update `data_out` from `data_in` every four `clk_en_in` in the following way; if `data_in` (coded as unsigned) is greater than 10 then `data_out` is equal to `data_in`, otherwise all bits of from `data_out` take the value zero. . In addition, a signal called `clk_en_out` must be high the first rising edge of `clk` after `data_out` is ready and must be low the next rising edge. The following `VHDL` code tries to do that.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity se_exam_2022 is
  port (clk       : in  std_logic; -- 3kHz
        clk_en_in : in  std_logic; -- 1kHz
        data_in   : in  std_logic_vector(3 downto 0);
```

[1] With a bandwidth less tan 1 kHz

```
        clk_en_out: out std_logic;
        data_out   : out std_logic_vector(3 downto 0)
        );
end entity;

architecture arch_1 of se_exam_2022 is
  signal n : unsigned(2 downto 0):= to_unsigned(0,3);
begin
  process(clk)
  begin
    if rising_edge(clk) then
      if clk_en_in = '1' then
        clk_en_out <='0';
        if n = 3 then
          if unsigned(data_in)>10 then
            clk_en_out <='1';
            data_out <= data_in;
            n <= to_unsigned(0,n'length);
          else
            data_out <= (others => '0');
          end if;
          n <= n+1;
        end if;
      end if;
    end if;
  end process;
end architecture;
```

1. Unfortunately, `clk_en_out` is not well generated. Draw the actual digital waveform (`clk`, `clk_en_in` and `clk_en_out`).

2. Modify the code and draw the right digital waveform.

3 FSM in VHDL (1 point)

1. Consider a state diagram (or graph) that describes a FSM. Can we say if we are dealing with a Moore or Mealy FSM just looking at the transition arc?

2. Draw the block diagram of a FSM.

3. Describe how to implement a FSM using different number of `process` and point out their differences.

4 Reading serial communication data with a RasPi (1point)

Consider the following code executed in a RasPi B+

```
  while 1:
  t0=time.time()
  x=ser.read(205)
  compute_something(x)
  t1=time.time()
  print t1-t0
```

1. Consider an Arduino sending 8-bit samples at $F_s = 8\,\text{kHz}$ through a 115 200 bps serial connection (with bit start and bit stop) and that the time to read 205 samples is 450 µs. The mean of the printed value $t_1 - t_0 = 25.625\,\text{ms}$. Can you determine the execution time of the *compute_something()* function? Can you limit its value?

2. Repeat the previous question considering that the mean of the printed value $t_1 - t_0 = 27\,\text{ms}$. What percentage of the samples are lost?

3. Repeat the two previous questions changing the baud rate from 115.2 kbps to 250 kbps.

5 Contact bounce (1 point)

We presented some *debouncing* solutions to the contact bounce problem inherent to switches.

1. Point out the different advantages and disadvantages between hardware and software solutions without describing any of them.

2. Some of these solutions needed to be tuned for each contact while others, the so called *universals* solutions, not. Enumerate the *universals* solutions and describe one of them in detail.